Pronet Series Servo Drive

Canopen User’s Manual

Estun Limited Warranty
This manual does not entitle you to any rights. Estun reserves the right to change this manual without prior notice. All rights reserved. The copyright is held by Estun. No part of this publication may be copied or reproduced without written permission from Estun.

—— content ——

Canopen User's Manual 1
1、general introduction 4
1.1 CAN main files 4
1.2 Terms and abbreviations used in this manual 4
1.3 Brief introduction of CANopen 5
3、CANopen Commuinication 8
3.1 CAN identifier list 10
3.2 SDO 11
3.3 PDO 14
3.3.1 PDO parameter 17
3.4 SYNC message 23
3.5 Emergency message 24
3.6 HEARTBEAT message 26
3.7 Network management (NMT service) 27
4、Conversion factors（Factor Group） 30
4.1 Objects treated in this chapter 31
4.1.1 position factor 31
4.1.2 velocity factor 33
4.1.3 acceleration factor 35
5、Position Control Function 36
5.1 Objects treated in this chapter 38
6、Device Control 40
6.1 State diagram (State machine) 40
6.2 controlword 42
6.2.1 controlword 43
6.2.2 statusword 44
6.2.3 shutdown_option_code 45
6.2.4 disable_operation_option_code 47
6.2.5 quick_stop_option_code 47
6.2.6 halt_option_code 49
6.2.7 fault_reaction_option_code 49
7、Control mode 50
7.1 Relevant parameter of control mode 50
7.1.1 modes_of_operation 50
7.1.2 modes_of_operation_display 51
7.2 HOMING MODE 52
7.2.1 Control word of homing mode. 52
7.2.2 Status word of homing mode 52
7.2.3 Relevant parameter of homing mode 54
7.2.4 Homing sequences 57
7.3 PROFILE VELOCITY MODE 59
7.3.1 Control word of profile velocity mode 59
7.3.2 Status word of velocity mode 59
7.3.3 Objects of profile velocity mode 59

7.4 PROFILE POSITION MODE 64
7.4.1 Control word of profile position mode 64
7.4.2 Status word of profile position mode 64
7.4.3 Objects of profile position mode 65
7.4.4 Functional Description 68
8、Parameters of the CAN interface 71
Appendix object dictionary 72

1. General introduction

1.1 CAN main files

Document Name	Source
CiA DS 301 V 4.01: CANopen Communication Profile for Industrial Systems - based on CAL	**CiA**
CiA DSP 402 V 2.0: CANopen Device Profile	**CiA**

1.2 Terms and abbreviations used in this manual

CAN	Controller Area Network
CiA	CAN in Automation International Users and Manufacturers Group.
COB	Communication Object; a unit of transportation on a CAN network. Data is sent across a network inside a COB. The COB itself is part of the CAN message frame.
EDS	Electronic Data Sheet; a node-specific ASCII-format file required when configuring the CAN network. The EDS file contains general information on the node and its dictionary objects (parameters).
LMT	Layer Management; one of the service elements of the CAN Application Layer in the CAN Reference Model. It serves to configure parameters for each layer in the CAN Reference Model.
NMT	Network Management; one of the service elements of the CAN Application Layer in the CAN Reference Model. It performs initialization, configuration and error handling on a CAN network.
OD	A local storage of all Communication Objects (COB) recognized by a device.。
Parameter	A parameter is an operating instruction for the drive. Parameters can be read and programmed with the drive control panel, or through the NCAN-02 Module.
PDO	Process Data Object; a type of COB. Used for transmitting time-critical data, such as control commands, references and actual values.
RO	Denotes read-only access.

RW Denotes read/write access.

SDO Service Data Object; a type of COB. Used for transmitting non-time critical data, such as parameters.

1.3 Brief introduction of CANopen

CANopen is a higher-layer protocol based on the CAN (Control Area Network) serial bus system and the CAL (CAN Application Layer). CANopen assumes that the hardware of the connected device has a CAN transceiver and a CAN controller as specified in ISO 11898.

The CANopen Communication Profile, CiA DS-301, includes both cyclic and event-driven communication, which makes it possible to reduce the bus load to minimum while still maintaining extremely short reaction times. High communication performance can be achieved at relatively low baud rates, thus reducing EMC problems and cable costs.

CANopen device profiles define both direct access to drive parameter and time-critical process data communication. The NCAN-02 fulfils CiA (CAN in Automation) standard DSP-402 (Drives and Motion Control), supporting the 'Manufacturer Specific' operating mode only.

The physical medium of CANopen is a differentially-driven two-wire bus line with common return according to ISO 11898. The maximum length of the bus is limited by the communication speed as follows:

Baud Rate	Max. Bus Length
1M bit/s	25 m
500k bit/s	100 m
250k bit/s	250 m
125k bit/s	500 m
100k bit/s	600 m
50k bit/s	1000 m

The maximum theoretical number of nodes is 127. However, in practice, the maximum number depends on the capabilities of the CAN transceivers used.

Further information can be obtained from the CAN in Automation International Users and Manufacturers Group (*www.can-cia.de*).

2. Cabling and Wiring

●The layout of CN3 terminal

Pin number	Name	Function
1	5V	5VDC power supply
2	5V	
3	485+	RS-485 communication terminal
4	DGND	Grounding
5	DGND	
6	485-	RS-485 communication terminal
7	**CANH**	**CAN communication terminal**
8	**CANL**	**CAN communication terminal**

● The layout of CN4 terminal

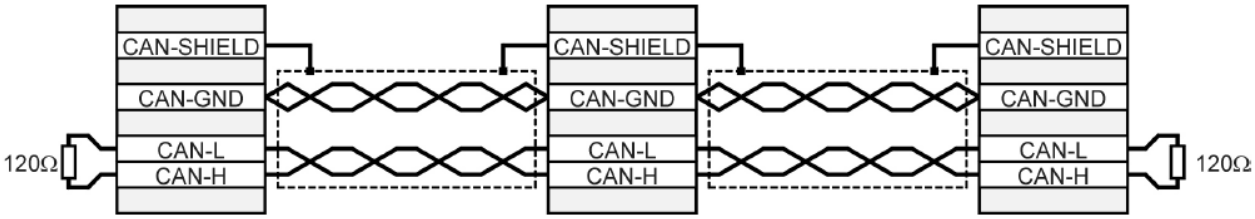
Pin number	Name	Function
1	—	Reserved
2	—	Reserved
3	485+	RS-485 communication terminal
4	DGND	Grounding
5	DGND	
6	485-	RS-485 communication terminal
7	**CANH**	**CAN communication terminal**
8	**CANL**	**CAN communication terminal**

CN3 is always the input terminal of communication cable and CN4 is always the output terminal of communication cable. (If connection to another communication node is necessary, the cable will connect CN4 to next communication node. If not, a terminal resistor could be applied at CN4). When multiple ProNet devices are connected, it is forbidden to connect the CN3 terminals of different drives directly.

For example, a network is composed of one PLC, three ProNet drives called A, B and C. The cabling network is as below:

PLC → CN3 of drive A，CN4 of drive A → CN3 of drive B，CN4 of drive B → CN3 of drive C, CN4 of drive C → 120 ．resistor。

The two ends of the CAN cable have to be terminated by a resistor of 120 ．+/- 5% as below..



Please select the bus cable with double twisted pair cables and shielding layer, one pair for connecting CAN-L and CAN-H, another pair for grounding.

3、CANopen Commuinication

CaL supplies all network management service and message transferring protocol with defining the content of object or type of object for communication. It defines how instead of what, which is the strength of CANopen.

CANopen is developed based on CAL. It applies CAL protocol subsets for communication and service and creates a solution to DCS. CANopen could freely extend the node function to simplicity or complex while the network nodes are accessible and available to each other.

The key concept of CANopen is object dictionary. This way of object description is also applied to other fieldbus system like Probis and Interbus-S. CANopen communication could access to all the parameter of drivers through object dictionary. Please notice object dictionary is not one part of CAL, instead of which it is realized in CANopne.

CANopen communication defines several types of objects as below…

Abbreviation	Full Spell	Description
SDO	**S**ervice **D**ata **O**bject	Used for normal parameterization of the servo controller
PDO	**P**rocess **D**ata **O**bject	Fast exchange of process data (e.g. velocity actual value) possible.
SYNC	**Sync**hronization Message	Synchronization of several CAN nodes
EMCY	**Em**ergen**cy** Message	Used to transmit error messages of the servo controller.
NMT	**N**etwork **M**anagemen**t**	Used for network services. For example the user can act on all controllers at the same time via this object type.
Heartbeat	Error Control Protocol	Used for observing all nodes by cyclic messages.

CAN employs data frames for transferring data between the host (controller) and the nodes on the bus. The following figure presents the structure of the data frame.

Start of frame	ARBITRATION FIELD		Control Field	Date Field	CYCLICAL REDUNDANCY CHECK	Acknowledged field	End of frame
	COB-ID	RTR					
1BIT	11 OR 29 BITS	1BIT	6BITS	0~8BYTES	16BITS	2BITS	7BITS

Our drivers doesn't support remote frame currently. The details of COB-ID is as below..

FUNCTION CODE				NODE ID						
10	9	8	7	6	5	4	3	2	1	0

3.1 CAN identifier list

Object	COB-ID bit10~7 （binary）	COB-ID (hex)	Index in OD
NMT	0000	000_h	—
SYNC	0001	080_h	1005_h、1006_h、1007_h
TIME STAMP	0010	100_h	1012_h、1013_h
EMCY	0001	081_h ~ $0FF_h$	1024_h、1015_h
PDO1（transimit）	0011	181_h ~ $1FF_h$	1800_h
PDO1（receive）	0100	201_h ~ $27F_h$	1400_h
PDO2（transimit）	0101	281_h ~ $2FF_h$	1801_h
PDO2（receive）	0110	301_h ~ $37F_h$	1401_h
SDO（transimit）	1011	581_h ~ $5FF_h$	1200_h
SDO（receive）	1100	601_h ~ $67F_h$	1200_h
Heartbeat	1110	701_h ~ $77F_h$	1016_h、1017_h

注意：

1、 PDO/SDO ‘s send/receive is observed by (slave) CAN.

2、 Our drive ‘s CANopen protocol currently supports 2 transimit PDO and 2 receive PDO.

3.2 SDO

SDO is used to visit the object dictionary of a device. Visitor is called client. The CANopen device whose object dictionary is visited and required to supply the asked service is called server. CANopen messages from a client and servo all contain 8 bits (Not all of them are meaningful). A request from a client must be confirmed by a server

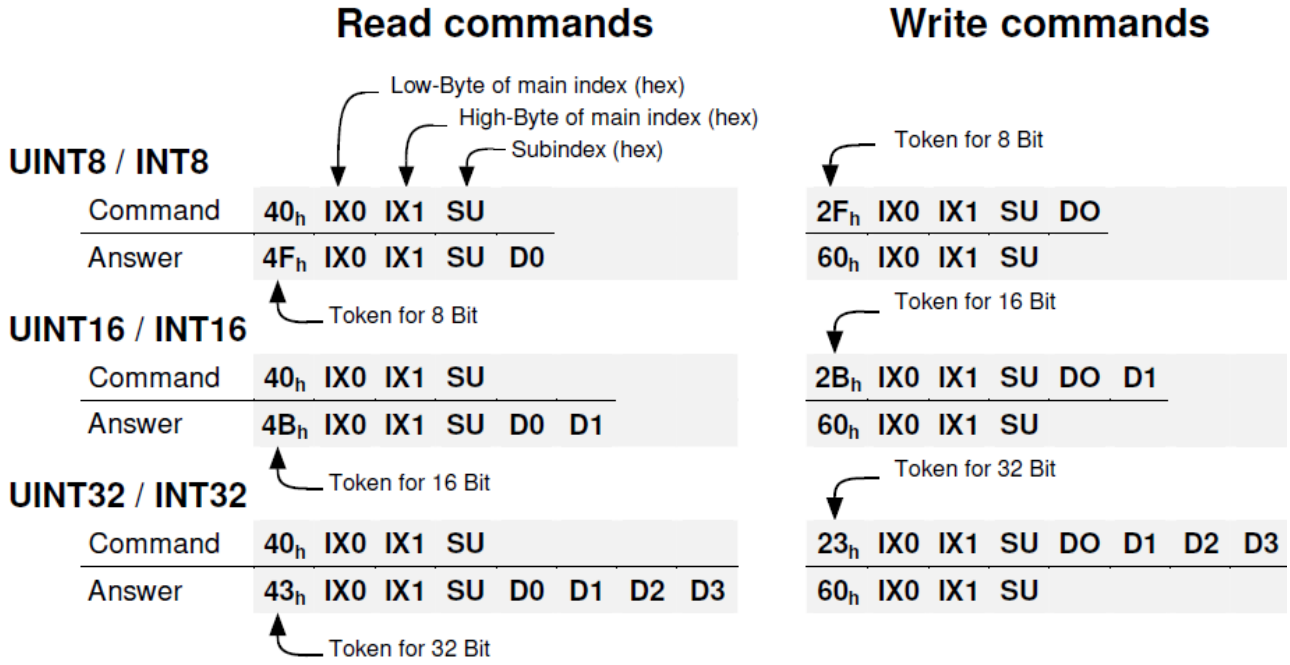
There are 2 method of convey for SDO：

- Expedited transfer ： contains 4 bytes at maximum
- Segmented transfer ： contains more than 4 bytes

Basic structure of SDO：

Byte0	Byte1~2	Byte3	Byte4~7
SDO	Object reference	Sub-object reference	data

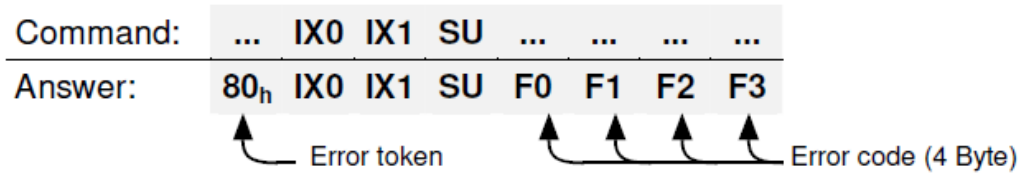
SDO read/write command structure:


Read commands
Write commands
Low-Byte of main index (hex)
High-Byte of main index (hex)
Subindex (hex)
UINT8 / INT8
Command 40h IX0 IX1 SU
Answer 4Fh IX0 IX1 SU D0
Token for 8 Bit
Token for 8 Bit
2Fh IX0 IX1 SU DO
60h IX0 IX1 SU
UINT16 / INT16
Command 40h IX0 IX1 SU
Answer 4Bh IX0 IX1 SU D0 D1
Token for 16 Bit
Token for 16 Bit
2Bh IX0 IX1 SU DO D1
60h IX0 IX1 SU
UINT32 / INT32
Command 40h IX0 IX1 SU
Answer 43h IX0 IX1 SU D0 D1 D2 D3
Token for 32 Bit
Token for 32 Bit
23h IX0 IX1 SU DO D1 D2 D3
60h IX0 IX1 SU

For example：

	Reading of Obj. 6061_00_h Returning data: 01_h	Writing of Obj. 1401_02_h Data: EF_h
UINT8 / INT8		
Command:	40_h 61_h 60_h 00_h	$2F_h$ 01_h 14_h 02_h EF_h
Answer:	$4F_h$ 61_h 60_h 00_h 01_h	60_h 01_h 14_h 02_h
UINT16 / INT16	Reading of Obj. 6041_00_h Returning data: 1234_h	Writing of Obj. 6040_00_h Data: $03E8_h$
Command:	40_h 41_h 60_h 00_h	$2B_h$ 40_h 60_h 00_h $E8_h$ 03_h
Answer:	$4B_h$ 41_h 60_h 00_h 34_h 12_h	60_h 40_h 60_h 00_h
UINT32 / INT32	Reading of Obj. 6093_01_h Returning data: 12345678_h	Writing of Obj. 6093_01_h Data: 12345678_h
Command:	40_h 93_h 60_h 01_h	23_h 93_h 60_h 01_h 78_h 56_h 34_h 12_h
Answer:	43_h 93_h 60_h 01_h 78_h 56_h 34_h 12_h	60_h 93_h 60_h 01_h

SDO-error messages：



Error code F3 F2 F1 F0	Description
05 03 00 00$_h$	Toggle bit not alternated
05 04 00 01$_h$	Client / server command specifier not valid or unknown
06 01 00 00$_h$	Unsupported access to an object
06 01 00 01$_h$	Attempt to read a write only object
06 01 00 02$_h$	Attempt to write a read only object
06 02 00 00$_h$	Object does not exist in the object dictionary
06 04 00 41$_h$	Object cannot be mapped to the PDO
06 04 00 42$_h$	The number and length of the objects to be mapped would exceed PDO length
06 04 00 47$_h$	General internal incompatibility in the device
06 07 00 10$_h$	Data type does not match, length of service parameter does not match
06 07 00 12$_h$	Data type does not match, length of service parameter too high
06 07 00 13$_h$	Data type does not match, length of service parameter too low
06 09 00 11$_h$	Sub-index does not exist
06 04 00 43$_h$	General parameter incompatibility
06 06 00 00$_h$	Access failed due to an hardware error *1)
06 09 00 30$_h$	Value range of parameter exceeded
06 09 00 31$_h$	Value of parameter written too high
06 09 00 32$_h$	Value of parameter written too low
06 09 00 36$_h$	Maximum value is less than minimum value
08 00 00 20$_h$	Data cannot be transferred or stored to the application *1)
08 00 00 21$_h$	Data cannot be transferred or stored to the application because of local control
08 00 00 22$_h$	Data cannot be transferred or stored to the application because of the present device state *3)
08 00 00 23$_h$	No Object Dictionary is present *2)

3.3 PDO

PDO is applied to transferring real time data which will be conveyed from a producer to one or multiple clients. Data transferring will be limited to 1 to 8 bytes. There is no hand-shake restriction in PDO communication, which means data has been redefined, so clients could process the received data for vary short time. PDO content will be only defined by its CAN ID, assuming producers and clients know PDO content from its CAN ID.

2 objects in object dictionary are used for each PDO.

- PDO communication parameter：It contains COB-ID, transferring type, restriction time and cycle of timer used by PDO.
- PDO mapping parameter：It contains a list of objects in the object dictionary. This objects are mapped into PDO, includes their data length in bits. Producers and clients must know this mapping to explain the content of PDO.

The content of PDO's message is predefined or configured when the network initializes. Mapping application object into PDO is described in object dictionary. If a device(producer and client) support dynamic ways, SDO could be used to configure PDO's mapping parameter. Our servo drive supports dynamic PDO mapping. There are 2 rules for PDO mapping to follow..

1、Each PDO could be mapped into 4 objects.

2、The length of each PDO will be no more than 64 bits.

PDO mapping process：

1、Set the sub-index of PDO coordinated mapping parameter(1600 $_h$,1601 $_h$,1A00 $_h$ or 1A01 $_h$) as o.

2、Revise the sub-index from 1 to 4 of PDO coordinated mapping parameter(1600 $_h$,1601 $_h$,1A00 $_h$ or 1A01 $_h$).

3、Set the sub-index 0 of PDO coordinated mapping parameter(1600 $_h$,1601 $_h$,1A00 $_h$ or 1A01 $_h$) as legal number(number of PDO's mapping objects)

4、PDO mapping completing.

There are multiple ways to transmit PDO:

- Synchronous（Synchronization by receiving SYNC object）

  Cycle: Transmit triggered after every 1 to 240 SYNC messages.

- Asynchronous

  Transmit triggered by special object event regulated in sub-object protocol.

Transmit type of PDO

Transmit Type	Description	PDO
0	Reserved	—
1~240	**SYNC**：It represents the number of SYNC objects between 2 PDOs.	TPDO/RPDO
240~253	Reserved	—
254	**Asynchronous**：If the content of PDO has changed, PDO transmit will be triggered.	TPDO
255	**Asynchronous**： The content of PDO will be periodically updated and transmitted.	TPDO/RPDO

One PDO could set a frozen time which is the shortest interval time between 2 continuous PDO. It could prevent the bus from being occupied by amount of data with high priority. Frozen time is defined by 16 bit unsigned integer number and its unit is 100us.

One PDO could set a timing period. When the regulated time is violated, a PDO transmit could be triggered without a trigger bit. Object timing period is defined as 16 bit unsigned integer and its unit is 1ms.

PDO mapping case：

Map the 3 objects to PDO1 (transmit). PDO1 (transmit) is required to be asynchronous periodic type with period time as much as 10ms and frozen time as much as 2ms.

Object	Index — Sub-index	Description
statusword	6041$_{h}$ – 00 $_{h}$	Status word
modes_of_operation_display	6061$_{h}$ – 00 $_{h}$	Practical operational mode
Position_Acture_Value	6064$_{h}$ – 00 $_{h}$	Practical position

1）、Clear number_of_mapped_objects

number_of_mapped_objects(10A0 $_{h}$： 00 $_{h}$)= 0

2）、Set the parameter for mapping objects

Index =6041 $_{h}$ Subin. = 00h Length = 10 $_{h}$ □ 1st_mapped_object(10A0 $_{h}$： 01 $_{h}$)= 60410010 $_{h}$

Index =6061 $_{h}$ Subin. = 00h Length = 08 $_{h}$ □ 2st_mapped_object(10A0 $_{h}$： 02 $_{h}$)= 60610008 $_{h}$

Index =60FD $_{h}$ Subin. = 00h Length = 20 $_{h}$ □ 3st_mapped_object(10A0 $_{h}$： 03 $_{h}$) = 60FD0020 $_{h}$

3）、Set number_of_mapped_objects

number_of_mapped_objects(10A0 $_{h}$： 00 $_{h}$)= 3

4）、Set PDO communication parameter

PDO1（transmit）is asynchronous periodical type □ transmission_type (1800 $_{h}$： 02 $_{h}$)= FF $_{h}$

Frozen time 2ms(20×100us) □ inhibit_time (10A0 $_{h}$： 03 $_{h}$)= 14 $_{h}$

Period time: 10ms(10×1ms) □ event_time (1800 $_{h}$： 05 $_{h}$)= 0A $_{h}$

5）、PDO mapping complete.

3.3.1 PDO parameter

PRONET drive contains 4 transmit PDOs and 4 receive PDOs. The detailed communication parameter and mapping parameter of the first transmit/receive PDO is as below and those of the rest 3 transmit/receive PDO are the same as the first PDO.

Index	1800 **h**
Name	transmit_pdo_parameter_tpdo1
Object Code	RECORD
No. of Elements	4

Sub-Index	01 **h**
Description	cob_id_used_by_pdo_tpdo1
Data Type	UINT32
Access	RW
PDO Mapping	NO
Units	——
Value Range	181 **h**...1FF **h**, Bit 31 may be set
Default Value	181 **h**

Sub-Index	02 **h**
Description	transmission_type_tpdo1
Data Type	UINT8
Access	RW
PDO Mapping	NO
Units	——
Value Range	1...240,254,255
Default Value	255

Sub-Index	03 **h**
Description	inhibit_time_tpdo1
Data Type	UINT16
Access	RW
PDO Mapping	NO
Units	100μs
Value Range	——
Default Value	100

Sub-Index	05 h
Description	event_time_tpdo1
Data Type	UINT16
Access	RW
PDO Mapping	NO
Units	1ms
Value Range	——
Default Value	10

Index	1A00 h
Name	transmit_pdo_mapping_tpdo1
Object Code	RECORD
No. of Elements	2

Sub-Index	00 h
Description	number_of_mapped_objects_tpdo1
Data Type	UINT8
Access	RW
PDO Mapping	NO
Units	——
Value Range	0...4
Default Value	2

Sub-Index	01 h
Description	first_mapped_object_tpdo1
Data Type	UINT32
Access	RW
PDO Mapping	NO
Units	——
Value Range	——
Default Value	See table

Sub-Index	02 h
Description	second_mapped_object_tpdo1
Data Type	UINT32
Access	RW
PDO Mapping	NO
Units	——
Value Range	——

Default Value	See table

Sub-Index	03 **h**
Description	third_mapped_object_tpdo1
Data Type	UINT32
Access	RW
PDO Mapping	NO
Units	——
Value Range	——
Default Value	See table

Sub-Index	04 **h**
Description	fourth_mapped_object_tpdo1
Data Type	UINT32
Access	RW
PDO Mapping	NO
Units	——
Value Range	——
Default Value	See table

1、T-PDO1

Index	Comment	Type	Acc.	Default Value
1800 h _00 h	number of entries	UINT8	RO	04 h
1800 h _01 h	COB-ID used by PDO	UINT32	RW	00000181 h
1800 h _02 h	transmission type	UINT8	RW	FF h
1800 h _03 h	inhibit time (100 μs)	UINT16	RW	64 h
1800 h _05 h	event time (1ms)	UINT16	RW	0A h
1A00 h _00 h	number of mapped objects	UINT8	RW	02 h
1A00 h _01 h	first mapped object	UINT32	RW	60410010 h
1A00 h _02 h	second mapped object	UINT32	RW	60640020 h
1A00 h _03 h	third mapped object	UINT32	RW	00 h
1A00 h _04 h	fourth mapped object	UINT32	RW	00 h

2、T-PDO2

Index	Comment	Type	Acc.	Default Value
1801 h _00 h	number of entries	UINT8	RO	04 h
1801 h _01 h	COB-ID used by PDO	UINT32	RW	00000281 h
1801 h _02 h	transmission type	UINT8	RW	FF h
1801 h _03 h	inhibit time (100 μs)	UINT16	RW	64 h
1801 h _05 h	event time (1ms)	UINT16	RW	0A h
1A01 h _00 h	number of mapped objects	UINT8	RW	02 h
1A01 h _01 h	first mapped object	UINT32	RW	60640020 h
1A01 h _02 h	second mapped object	UINT32	RW	60610010 h
1A01 h _03 h	third mapped object	UINT32	RW	00 h
1A01 h _04 h	fourth mapped object	UINT32	RW	00 h

3、T-PDO3

Index	Comment	Type	Acc.	Default Value
1802 h _00 h	number of entries	UINT8	RO	04 h
1802 h _01 h	COB-ID used by PDO	UINT32	RW	00000281 h
1802 h _02 h	transmission type	UINT8	RW	FF h
1802 h _03 h	inhibit time (100 μs)	UINT16	RW	64 h
1802 h _05 h	event time (1ms)	UINT16	RW	0A h
1A02 h _00 h	number of mapped objects	UINT8	RW	02 h
1A02 h _01 h	first mapped object	UINT32	RW	60640020 h
1A02 h _02 h	second mapped object	UINT32	RW	60610010 h
1A02 h _03 h	third mapped object	UINT32	RW	00 h
1A02 h _04 h	fourth mapped object	UINT32	RW	00 h

4、T-PDO4

Index	Comment	Type	Acc.	Default Value
1803 h _00 h	number of entries	UINT8	RO	04 h
1803 h _01 h	COB-ID used by PDO	UINT32	RW	00000281 h
1803 h _02 h	transmission type	UINT8	RW	FF h
1803 h _03 h	inhibit time (100 μs)	UINT16	RW	64 h
1803 h _05 h	event time (1ms)	UINT16	RW	0A h
1A03 h _00 h	number of mapped objects	UINT8	RW	02 h
1A03 h _01 h	first mapped object	UINT32	RW	60640020 h
1A03 h _02 h	second mapped object	UINT32	RW	60610010 h
1A03 h _03 h	third mapped object	UINT32	RW	00 h
1A03 h _04 h	fourth mapped object	UINT32	RW	00 h

tpdo_1_transmit_mask

Index	Comment	Type	Acc.	Default Value
2000 h _00 h	number of entries	UINT8	RO	02 h
2000 h _01 h	tpdo_1_transmit_mask_low	UINT32	RW	FFFFFFFF h
2000 h _02 h	tpdo_1_transmit_mask_high	UINT32	RW	FFFFFFFF h

tpdo_2_transmit_mask

Index	Comment	Type	Acc.	Default Value
2001 h _00 h	number of entries	UINT8	RO	02 h
2001 h _01 h	tpdo_2_transmit_mask_low	UINT32	RW	FFFFFFFF h
2001 h _02 h	tpdo_2_transmit_mask_high	UINT32	RW	FFFFFFFF h

tpdo_3_transmit_mask

Index	Comment	Type	Acc.	Default Value
2002 h _00 h	number of entries	UINT8	RO	02 h
2002 h _01 h	tpdo_1_transmit_mask_low	UINT32	RW	FFFFFFFF h
2002 h _02 h	tpdo_1_transmit_mask_high	UINT32	RW	FFFFFFFF h

tpdo_4_transmit_mask

Index	Comment	Type	Acc.	Default Value
2003 h _00 h	number of entries	UINT8	RO	02 h
2003 h _01 h	tpdo_2_transmit_mask_low	UINT32	RW	FFFFFFFF h
2003 h _02 h	tpdo_2_transmit_mask_high	UINT32	RW	FFFFFFFF h

1、R-PDO1

Index	Comment	Type	Acc.	Default Value
1400 h _00 h	number of entries	UINT8	RO	02 h
1400 h _01 h	COB-ID used by PDO	UINT32	RW	00000201 h
1400 h _02 h	transmission type	UINT8	RW	FF h
1600 h _00 h	number of mapped objects	UINT8	RW	02 h
1600 h _01 h	first mapped object	UINT32	RW	60400010 h
1600 h _02 h	second mapped object	UINT32	RW	60FF0020 h
1600 h _03 h	third mapped object	UINT32	RW	00 h
1600 h _04 h	fourth mapped object	UINT32	RW	00 h

2、R-PDO2

Index	Comment	Type	Acc.	Default Value
1401 h _00 h	number of entries	UINT8	RO	02 h
1401 h _01 h	COB-ID used by PDO	UINT32	RW	00000301 h
1401 h _02 h	transmission type	UINT8	RW	FF h
1601 h _00 h	number of mapped objects	UINT8	RW	02 h
1601 h _01 h	first mapped object	UINT32	RW	60FF0020 h
1601 h _02 h	second mapped object	UINT32	RW	60600010 h
1601 h _03 h	third mapped object	UINT32	RW	00 h
1601 h _04 h	fourth mapped object	UINT32	RW	00 h

3、R-PDO3

Index	Comment	Type	Acc.	Default Value
1402 h _00 h	number of entries	UINT8	RO	02 h
1402 h _01 h	COB-ID used by PDO	UINT32	RW	00000301 h
1402 h _02 h	transmission type	UINT8	RW	FF h
1602 h _00 h	number of mapped objects	UINT8	RW	02 h
1602 h _01 h	first mapped object	UINT32	RW	60FF0020 h
1602 h _02 h	second mapped object	UINT32	RW	60600010 h
1602 h _03 h	third mapped object	UINT32	RW	00 h
1602 h _04 h	fourth mapped object	UINT32	RW	00 h

4、R-PDO4

Index	Comment	Type	Acc.	Default Value
1403 h _00 h	number of entries	UINT8	RO	02 h
1403 h _01 h	COB-ID used by PDO	UINT32	RW	00000301 h
1403 h _02 h	transmission type	UINT8	RW	FF h
1603 h _00 h	number of mapped objects	UINT8	RW	02 h
1603 h _01 h	first mapped object	UINT32	RW	60FF0020 h
1603 h _02 h	second mapped object	UINT32	RW	60600010 h
1603 h _03 h	third mapped object	UINT32	RW	00 h
1603 h _04 h	fourth mapped object	UINT32	RW	00 h

3.4 SYNC message

Network synchronization: The current input will be preserved and transmitted (if necessary) in the whole network. Output value will be updated by previous SYNC message.

Client-server mode:

CANopen suggests a COB-ID with highest priority to ensure that synchronized signal could be transmitted properly. Without transferring data, SYNC message could be as short as possible.

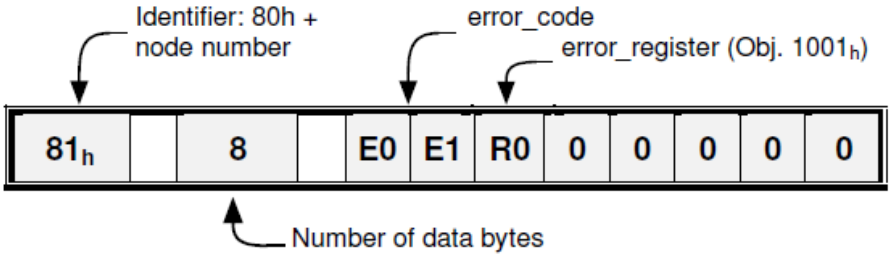
The identifier the servo controller receives SYNC messages is fixed to 080h. The identifier
can be read via the object **cob_id_sync**.。

Index	1005 **h**
Name	cob_id_sync
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	NO
Units	——
Value Range	80000080 **h**, 00000080 **h**
Default Value	00000080 **h**

3.5 Emergency message

When an alarm occurs to drive, CANopen will initiate an Emergency message to inform the current drive type and error code to clients.

The structure of Emergency message：



Alarm code

error_code（hex）	Description
2310	Over current
3100	Instantaneous power failure
3110	Over voltage
3120	Under voltage
5080	RAM exception
5210	AD sampling error
5420	Regenerative resistor error
5421	Regenerative resistor exception
5581	Parameter checksum exception
5582	electric gear error
5583	Motor type or drive type error
6100	Illegal error code
6120	PDO mapping error
6300	CAN communication error(Address or communication baud rate error)
7303	serial encoder error
7305	Incremental encoder error
7380	Resolver error
8100	CAN communication exception
8110	CAN bus overflow
8120	PASSIVE CAN bus turn to PASSIVE
8130	Heartbeat error
8140	CAN BUS OFF
8200	Length of CAN messages error
8210	Length of receiving PDO error
8311	Overload alarm
8480	Over speed alarm

相关参数

Index	1003 h
Name	pre_defined_error_field
Object Code	ARRAY
No. of Elements	4
Data Type	UINT32

Sub-Index	01 h
Description	standard_error_field_0
Access	RO
PDO Mapping	NO
Units	——
Value Range	——
Default Value	——

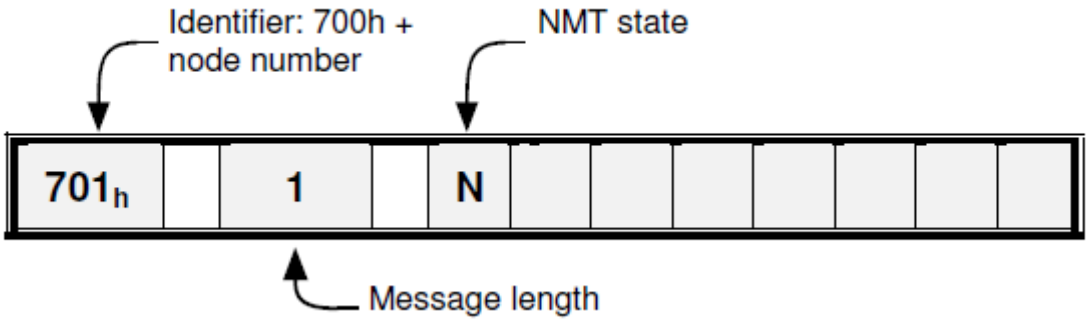
Sub-Index	02 h
Description	standard_error_field_1
Access	RO
PDO Mapping	NO
Units	——
Value Range	——
Default Value	——

Sub-Index	03 h
Description	standard_error_field_2
Access	RO
PDO Mapping	NO
Units	——
Value Range	——
Default Value	——

Sub-Index	04 h
Description	standard_error_field_3
Access	RO
PDO Mapping	NO
Units	——
Value Range	——
Default Value	——

3.6 HEARTBEAT message

Structure of the heartbeat message：

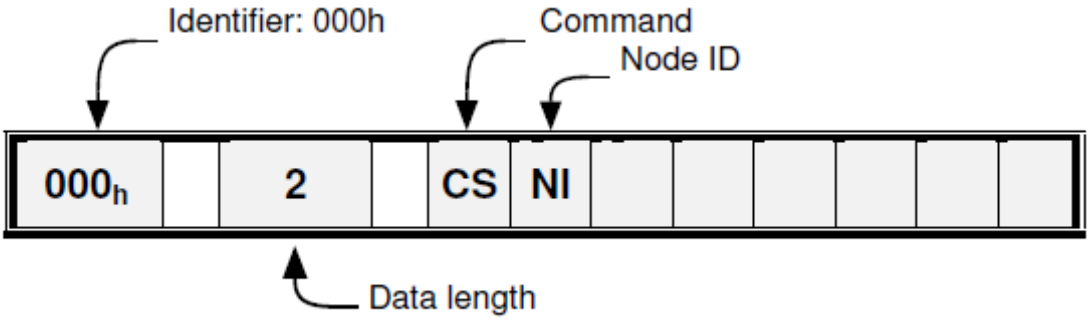


Relevant parameter：

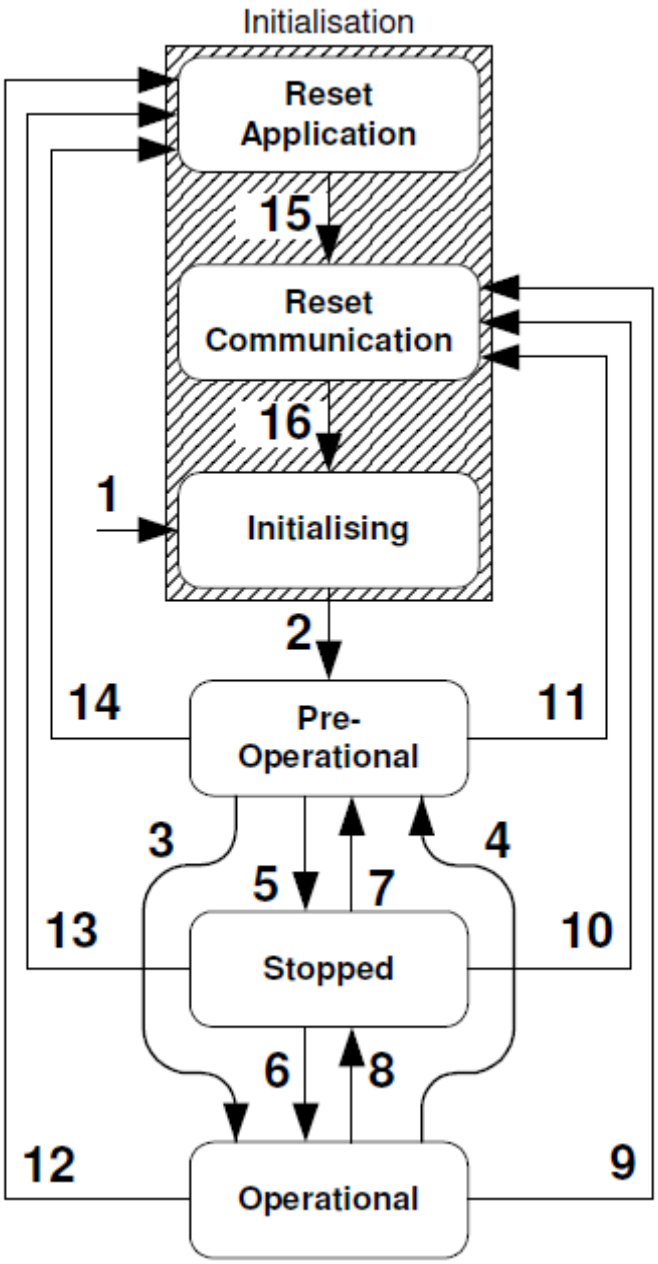
Index	1017 h
Name	producer_heartbeat_time
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	NO
Units	ms
Value Range	0 - 65535
Default Value	0

3.7 Network management (NMT service)

Structure of the message：



NMT-State machine：

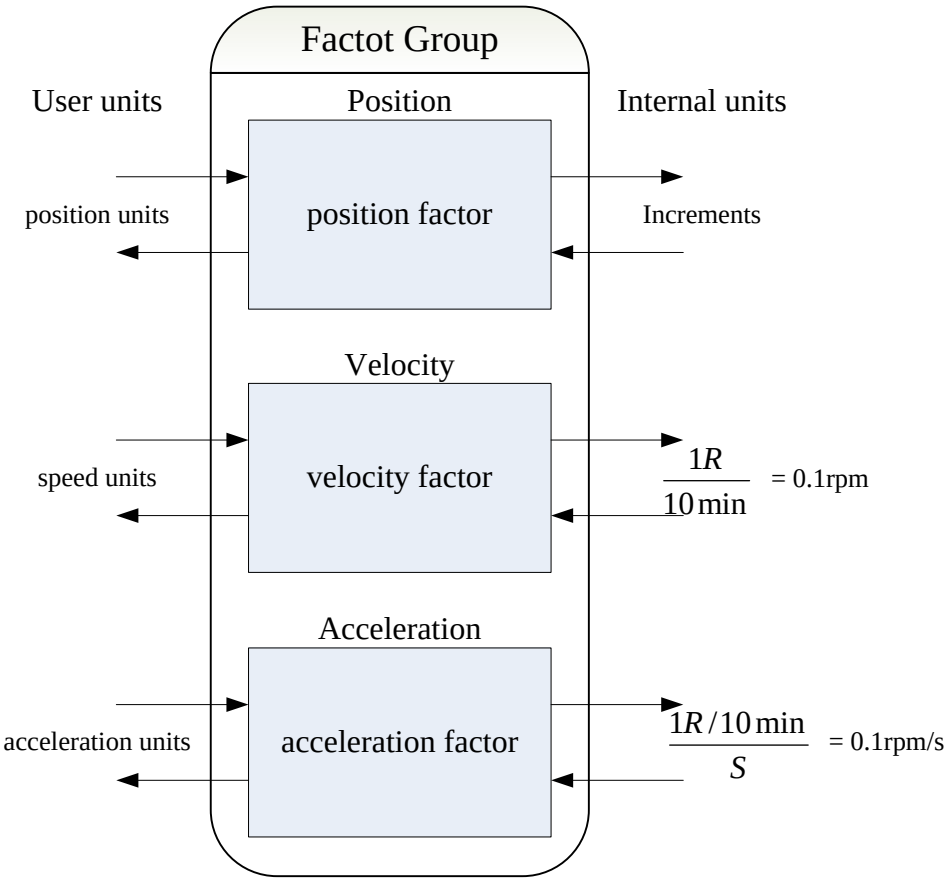


CS	Meaning	Transition	Target state
01_h	Start Remote Node	3, 6	**Operational**
02_h	Stop Remote Node	5, 8	**Stopped**
80_h	Enter Pre-Operational	4, 7	**Pre-Operational**
81_h	Reset Application	12, 13, 14	**Reset Application**
82_h	Reset Communication	9, 10, 11	**Reset Communication**

Name	Meaning	SDO	PDO	NMT
Reset Application	No communication. All CAN objects are set to their reset values (application parameter set).	-	-	-
Reset Communication	No communication. The CAN controller will be re-initialised.	-	-	-
Initialising	State after Hardware Reset. Reset of the CAN node, sending of the Bootup message	-	-	-
Pre-Operational	Communication via SDOs possible. PDOs inactive (No sending / receiving)	X	-	X
Operational	Communication via SDOs possible. PDOs active (sending / receiving)	X	X	X
Stopped	No communication except heartbeat + NMT	-	-	X

4、Conversion factors（**Factor Group**）

Servo controllers will be used in a huge number of applications: As direct drive, with gear or for linear drives. To allow an easy parameterization for all kinds of applications, the servo controller can be parameterized in such a way that all values like the demand velocity refer to the driven side of the plant. The necessary calculation is done by the servo controller.



The default setting of the Factor Group is as follows:

Value	Name	Unit	Remark
Length	position units	Increments	Increments per revolution *
Velocity	speed units	1R /10min	0.1rpm
Acceleration	Acceleration units	1R/10min/s	0.1rpm/s

***：** Common incremental encoder: 10000P/R
Resolver: 65536P/R
17 bit incremental encoder: 131072P/R
17 bit absolute encoder: 131072P/R

4.1 Objects treated in this chapter

Index	Object	Name	Type	Attr.
6093 h	ARRAY	position factor	UINT32	RW
6094 h	ARRAY	velocity factor	UINT32	RW
6097 h	ARRAY	acceleration factor	UINT32	RW

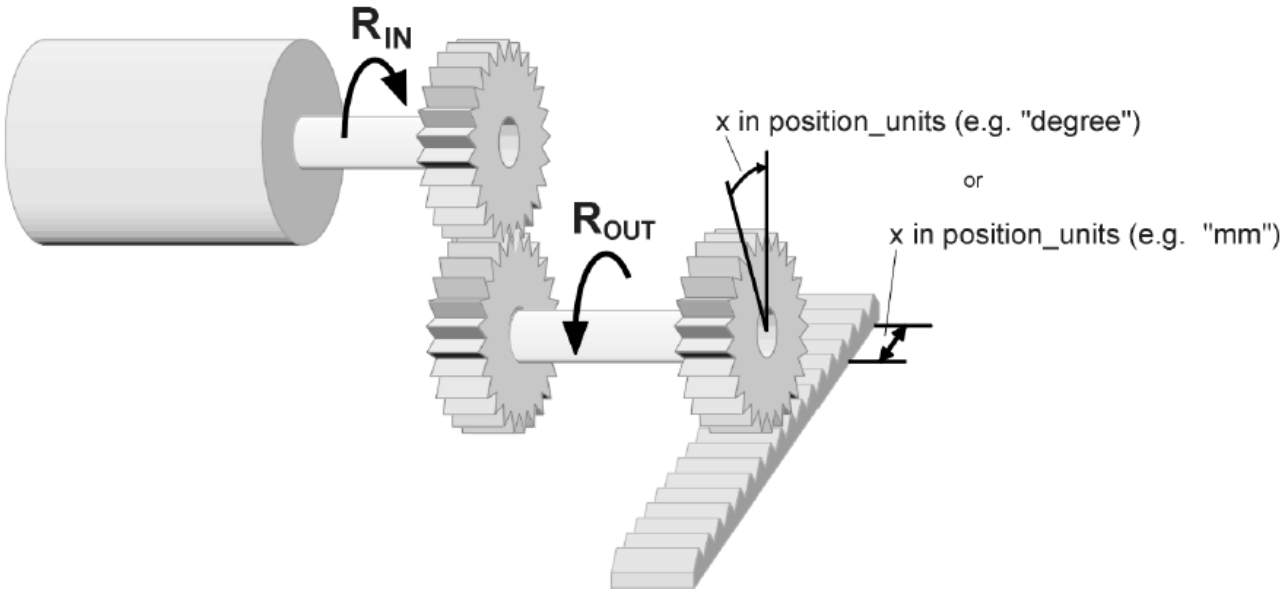
4.1.1 position factor

The object **position factor** converts all values of length of the application from **Position units** into the internal unit **increments** (65536 Increments equals 1 Revolution). It consists of numerator and divisor:

Index	6093 h
Name	position factor
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	01 h
Description	numerator
Access	RW
PDO Mapping	YES
Units	——
Value Range	——
Default Value	When power on, this value will be initiated to parameter Pn201

Sub-Index	02 h
Description	division
Access	RW
PDO Mapping	YES
Units	——
Value Range	——
Default Value	When power on, this value will be initiated to parameter Pn202



To calculate the **position factor** the following values are necessary:

gear_ratio Ratio between revolutions on the driving side (R_{IN}) and revolutions on the driven side (R_{OUT}).

feed_constant Ratio between revolutions on the driven side (R_{OUT}) and equivalent motion in **position_units** (e.g. 1 rev = 360°)

The calculation of the **position_factor** is done with the following equation:

$$\textbf{position factor} = \frac{\text{numerator}}{\text{division}} = \frac{\text{gear\_ratio} * \text{encoder\_resolution}}{\text{feed\_constant}}$$

注：

Encoder type	encoder_resolution(Unit: Inc)
Common incremental encoder	10000
Resolver	65535
17 bit incremental encoder	131072
17 bit absolute encoder: 131072P/R	131072

4.1.2 Velocity factor

The object **velocity factor** converts all speed values of the application from **speed_units** into the internal unit **revolutions per 4096 minutes**. It consists of numerator and divisor

Index	6094 **h**
Name	velocity factor
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	01 **h**
Description	numerator
Access	RW
PDO Mapping	YES
Units	——
Value Range	——
Default Value	1

Sub-Index	02 **h**
Description	division
Access	RW
PDO Mapping	YES
Units	——
Value Range	——
Default Value	1

In principle the calculation of the **velocity factor** is composed of two parts: A conversion factor from internal units of length into **position_units** and a conversion factor from internal time units into user defined time units (e.g. from seconds to minutes). The first part equals the calculation of the **position_factor.** For the second part another factor is necessary for the calculation:

time_factor_v Ratio between internal and user defined time units. (z.B. **1** ***min*** **=** $^{1}/_{10}$ ***10 min***)

gear_ratio Ratio between revolutions on the driving side (R_{IN}) and revolutions on the driven side (R_{OUT}).

feed_constant Ratio between revolutions on the driven side (R_{OUT}) and equivalent motion in position_units (e.g. 1 R = 360°)

The calculation of the **velocity factor** is done with the following equation:

$$\textbf{velocity factor} = \frac{\text{numerator}}{\text{division}} = \frac{\text{gear\_ratio} * \text{time\_factor\_v}}{\text{feed\_constant}}$$

4.1.3 acceleration factor

The object **acceleration_factor** converts all acceleration values of the application from **acceleration_units** into the internal unit（0.1rpm）。It consists of numerator and divisor:

Index	6094 **h**
Name	acceleration factor
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	01 **h**
Description	numerator
Access	RW
PDO Mapping	YES
Units	——
Value Range	——
Default Value	1

Sub-Index	02 **h**
Description	division
Access	RW
PDO Mapping	YES
Units	——
Value Range	——
Default Value	1

The calculation of the **acceleration_factor** is also composed of two parts: A conversion factor from internal units of length into **position_units** and a conversion factor from internal time units squared into user defined time units squared (e.g. from seconds^2 to minutes^2). The
first part equals the calculation of the **position_factor.** For the second part another factor is necessary for the calculation

time_factor_a Ratio between internal time units squared and user defined time units squared
（z.B.： 1**min**^{2} = 1min*min = 60s*1min =60/10 **10min/s**）

gear_ratio Ratio between revolutions on the driving side (R_{IN}) and revolutions on the driven side (R_{OUT}).

feed_constant Ratio between revolutions on the driven side (R_{OUT}) and equivalent motion in position_units (e.g. 1 R = 360°)

The calculation of the **acceleration_factor** is done with the following equation:

$$\textbf{acceleration factor} = \frac{\text{numerator}}{\text{division}} = \frac{\text{gear\_ratio} * \text{time\_factor\_a}}{\text{feed\_constant}}$$

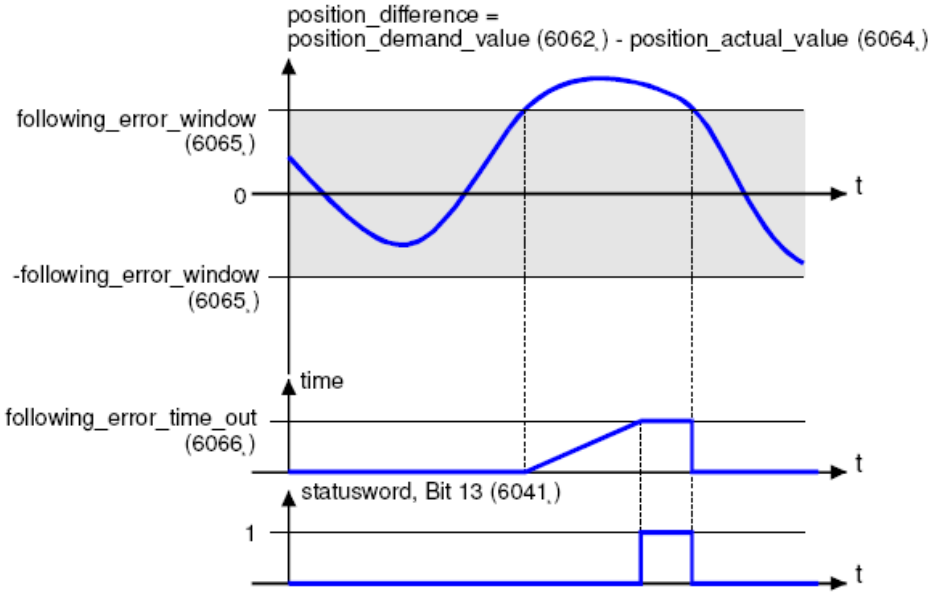
5、Position Control Function

This chapter describes all parameters which are required for the position controller. The desired position value (**position_demand_value**) of the trajectory generator is the input of the position controller. Besides this the actual position value (**position_actual_value**) is supplied by the angle encoder (resolver, incremental encoder, etc.). The behaviour of the position controller can be influenced by parameters.

It is possible to limit the output quantity (**control_effort**) in order to keep the position control system stable. The output quantity is supplied to the speed controller as desired speed value. In the **Factor Group** all input and output quantities are converted from the application-specific units to the respective internal units of the controller

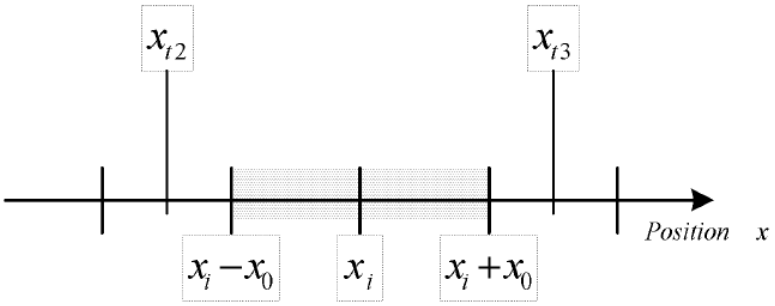
The following subfunctions are defined in this chapter:

1. Trailing error (Following Error)



Trailing error (Following Error) – Function Survey

The deviation of the actual position value (**position_actual_value**) from the desired position value (**position_demand_value)** is named trailing error. If for a certain period of time this trailing error is bigger than specified in the trailing error window (**following_error_window**) bit 13 (**following_error**) of the object **statusword** will be set to 1.

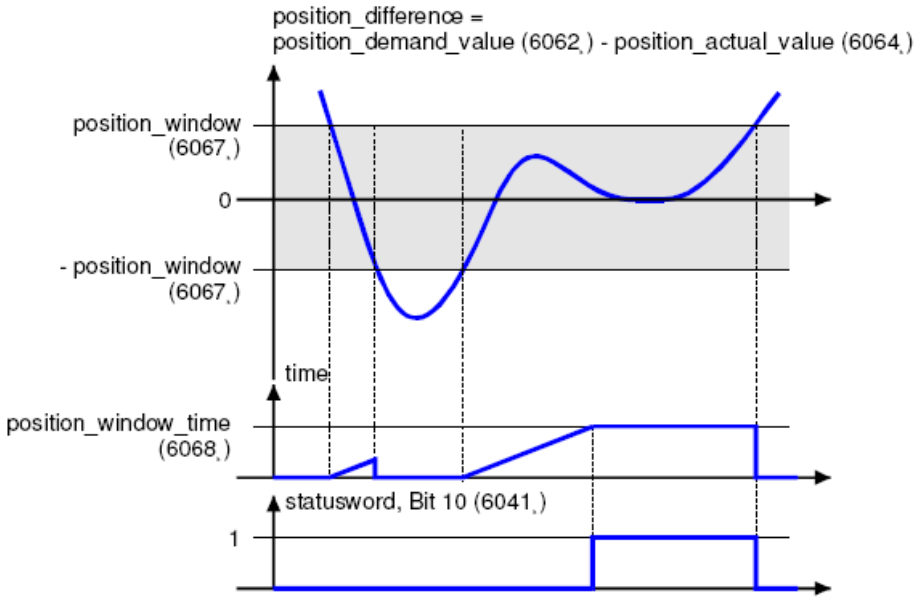


Trailing error (following error)

The permissible time can be defined via the object **following_error_time_out**.。 Figure above shows how the window function is defined for the message "following error".The range between xi-x0 and xi+x0 is defined symmetrically around the desired position (**position_demand_value**) xi. For example the positions xt2 and xt3 are outside this window (**following_error_window**). If the drive leaves this window and does not return to the window within the time defined in the object **following_error_time_out** then bit 13 (**following_error**) in the **statusword** will be set to 1.

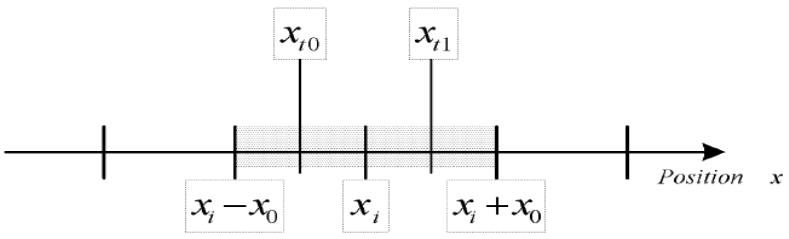
2**.** Position Reached

This function offers the chance to define a position window around the target position (**target_position**). If the actual position of the drive is within this range for a certain period of time – the **position_window_time** – bit 10 (**target_reached**) will be set to 1 in the statusword.



Position reached-function description

Figure below shows how the window function is defined for the message ”position reached”. The position range between xi-x0 and xi+x0 is defined symmetrically around the target position (**target_position**) xi. For example the positions xt0 and xt1 are inside this position window (**position_window**). If the drive is within this window a timer is started. If this timer reaches the time defined in the object **position_window_time** and the drive uninterruptedly was within the valid range between xi-x0 and xi+x0, bit 10 (**target_reached**) will be set in the **statusword**. As far as the drive leaves the permissible range, bit 10 is cleared and the timer is set to zero.



Position reached

5.1 Objects treated in this chapter

Index	Object	Name	Type	Attr.
6062 h	VAR	position_demand_value	INT32	RO
6063 h	VAR	position_actual_value*	INT32	RO
6064 h	VAR	position_actual_value	INT32	RO
6065 h	VAR	following_error_window	UINT32	RW
6066 h	VAR	following_error_time_out	UINT16	RW
6067 h	VAR	position_window	UINT32	RW
6068 h	VAR	position_time	UINT16	RW
60FA h	VAR	control_effort	INT32	RO

Index	6062 h
Name	position_demand_value
Object Code	VAR
Data Type	INT32
Access	RO
PDO Mapping	YES
Units	position units
Value Range	--
Default Value	--

Index	6064 h
Name	position_ actual _value
Object Code	VAR
Data Type	INT32
Access	RO
PDO Mapping	YES
Units	position units
Value Range	--
Default Value	--

Index	6065 h
Name	following_error_window
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	position units
Value Range	0 – 7FFFFFFF h
Default Value	256

Index	6066 h
Name	following_error_time_out
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	ms
Value Range	0 – 65535
Default Value	0

Index	60FA h
Name	control_effort
Object Code	VAR
Data Type	INT32
Access	RO
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	--

Index	6067 h
Name	position_window
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	position units
Value Range	--
Default Value	400

Index	6068 h
Name	position_time
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	ms
Value Range	0 – 65535
Default Value	0

6、Device Control

The following chapter describes how to control the servo controller using CANopen, i.e. how to switch on the power stage or to reset an error.

6.1 State diagram (State machine)

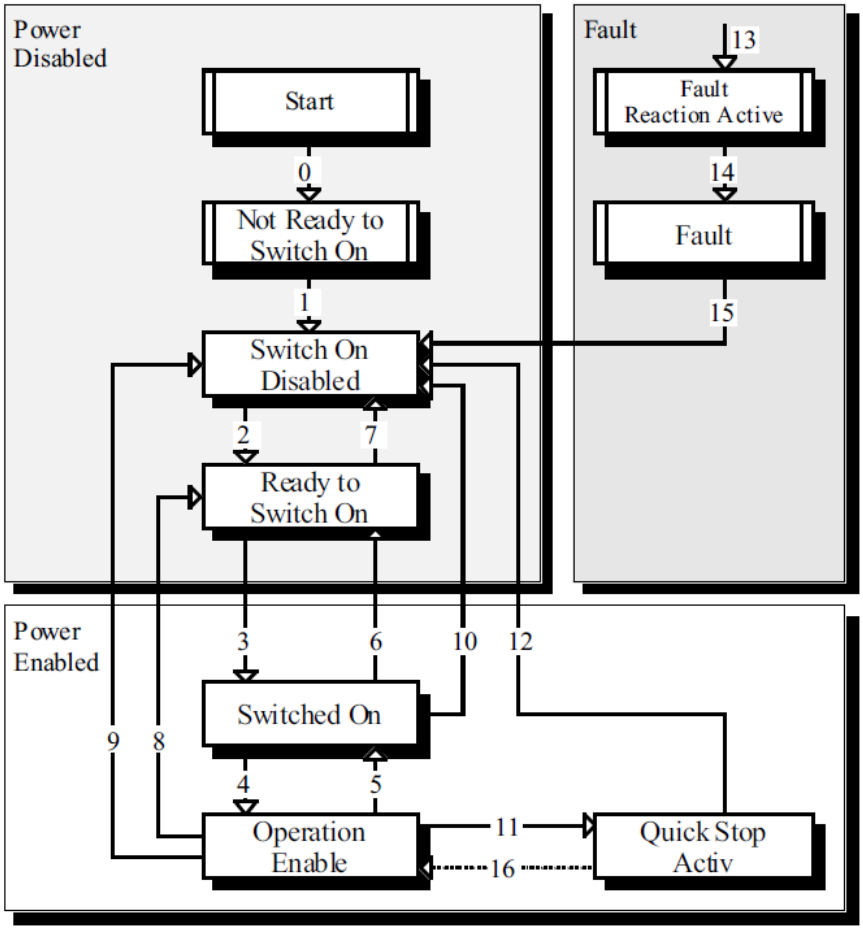
Using CANopen the complete control of the servo is done by two objects. Via the **controlword** the host is able to control the servo, as the status of the servo can be read out of the **statusword**. The following items will be used in this chapter:

State： The servo controller is in different states dependent on for instance if the power stage is alive or if an error has occurred. States defined under CANopen will be explained in this chapter.
Example: **SWITCH_ON_DISABLED**

State Transition： Just as the states it is defined as well how to move from one state to another (e.g. to reset an error). These state transitions will be either executed by the host by setting bits in the **controlword** or by the servo controller itself, if an error occurs for instance.

Command： To initiate a state transition defined bit combinations have to be set in the **controlword**. Such bit combination are called command. Example: **Enable Operation**

State diagram： All the states and all state transitions together form the so called state diagram: A survey of all states and the possible transitions between two states.。



State diagram of the servo controller

The state diagram can be divided into three main parts: "Power Disabled" means the power stage is switched off and "Power Enabled" the power stage is live. The area "Fault" contains
all states necessary to handle errors of the controller. The most important states have been highlighted in the Figure: After switching on the servo controller initializes itself and reaches the state **SWITCH_ON_DISABLED** after all. In this state CAN communication is possible and the servo controller can be parameterized (e.g. the mode of operation can be set to "velocity control"). The power stage remains switched off and the motor shaft is freely rotatable. Through the state transitions 2, 3 and 4 – principally like the controller enable under CANopen - the state **OPERATION_ENABLE** will be reached. In this state the power stage is live and the servo controller controls the motor according to the parameterized mode of operation. Therefore previously ensure that the servo controller has been parameterized correctly and the according demand value is zero. The state transition 9 complies with disabling the power stage, i.e. the motor is freely rotatable.

Status	Description
Not Ready to Switch On	The servo controller executes its self-test. The CAN communication is not working
Switch On Disabled	The self-test has been completed. The CAN communication is activated..
Ready to Switch On	Servo driver is waiting for the state of Switch and servo motor is not at power stage
Switched On	The power stage is alive.
Operation Enable	The motor is under voltage and is controlled according to operational mode

Quick Stop Active	Servo driver will be stopped through its fixed way,
Fault Reaction Active	Servo driver tests error and will be stopped through its fixed way, with motor's power stage alive
Fault	An error has occurred. The power stage has been switched off.

6.2 controlword

Index	Object	Name	Type	Attr.
6040 $_{h}$	VAR	controlword	UINT16	RW
6041 $_{h}$	VAR	statusword	UINT16	RO
605A $_{h}$	VAR	quick_stop_option_code	INT16	RW
605B $_{h}$	VAR	shutdown_option_code	INT16	RW
605C $_{h}$	VAR	disabled_operation_option_code	INT16	RW
605D $_{h}$	VAR	halt_option_code	INT16	RW
605E $_{h}$	VAR	fault_reaction_option_code	INT16	RW

6.2.1 controlword

Index	6040 h
Name	controlword
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	--
Value Range	--
Default Value	0

Controlword bit description is as below…

15 11	10 9	8	7	6 4	3	2	1	0
manufacturer specific	reserved	halt	Fault reset	Operation mode specific	Enable operation	Quick stop	Enable voltage	Switch on

Bit0 ~ 3 和 Bit7：

Transmit of status machine is triggered by 5 bits coordinated control code as below…

Command	Bit of the *controlword*					Transitions
	Fault reset	Enable operation	Quick stop	Enable voltage	Switch on	
Shutdown	0	X	1	1	0	2,6,8
Switch on	0	0	1	1	1	3*
Switch on	0	1	1	1	1	3**
Disable voltage	0	X	X	0	X	7,9,10,12
Quick stop	0	X	0	1	X	7,10,11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4,16
Fault reset	↑	X	X	X	X	15

Device control list

Notice: X means this bit could be ignored.

Bit4、5、6、8：

The definition of this 4 bit is different in different control mode…

Bit	Control Mode		
	profile position mode	profile velocity mode	homing mode
4	new_set_point	reserved	start_homeing_operation
5	change_set_immediatly	reserved	reserved
6	abs/rel	reserved	reserved
8	Halt	Halt	Halt

Other bits: all reserved

6.2.2 statusword

Index	6041 h
Name	statusword
Object Code	VAR
Data Type	UINT16
Access	RO
PDO Mapping	YES
Units	--
Value Range	--
Default Value	--

Explanation of statusword bit is as below:

bit	name
0	Ready to switch on
1	Switched on
2	Operation enabled
3	Fault
4	Voltage enabled
5	Quick stop
6	Switch on disabled
7	Warning
9~8	Not used now
10	Target reached
11	Internal limit active
13~12	Operation mode specific
15~14	Not used now

Bit0 ~ 3 、Bit5 和 Bit6：

The combination of these bit indicates the status of drives.

Value (binary)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

Bit4： Voltage enabled

Main power is on when this bit is 1.

Bit5： Quick stop

Driver will follow setting （605A $_h$：quick_stop_option_code）to halt when this bit is 0.

Bit7： Warning

Driver detects alarm when this bit is 1.

Bit10： Target reached

In different control modes the meaning of this bit is different.

In profile position mode, when set position is reached, this bit is set. When Halt is booted, speed is reduced to 0 and this bit will be set. When new position is set, this bit will be cleared.

In profile Velocity Mode, when the speed reaches the targeted speed, this bit will be set. When Halt is booted and speed is reduced to 0, this bit is set.

Bit11： Internal limit active

When this bit is 1, it indicates that internal torque has surpassed the set value.

Bit12、13：

These 2 bits mean different in different control mode…

Bit	Control mode		
	profile position mode	profile velocity mode	homing mode
12	Set-point acknowledge	Speed	Homing attained
13	Following error	Max slippage error	Homing error

Other bits：

All reserved.

6.2.3 shutdown_option_code

The object **shutdown_option_code** determines the behaviour if the state transition 8 (from **OPERATION ENABLE** to **READY TO SWITCH ON)** will be executed..

Index	605B $_h$
Name	shutdown_option_code
Object Code	VAR
Data Type	INT16
Access	RW
PDO Mapping	NO
Units	--
Value Range	0,1
Default Value	0

Value	Name
0	Power stage will be switched off. Motor is freely rotatable.。
1	Switch off the power stage after the motor stops deceleration..

6.2.4 disable_operation_option_code

The object **disable_operation_option_code** determines the behaviour if the state transition 5 (from **OPERATION ENABLE** to **SWITCHED ON)** will be executed.

Index	605C $_{h}$
Name	disable_operation_option_code
Object Code	VAR
Data Type	INT16
Access	RW
PDO Mapping	NO
Units	--
Value Range	0,1
Default Value	0

value	Desciption
0	Power stage will be switched off. Motor is freely rotatable.。
1	Switch off the power stage after the motor stops deceleration.

6.2.5 quick_stop_option_code

The object **quick_stop_option_code** determines the behaviour if a **Quick Stop** will be executed..

Index	605A $_{h}$
Name	quick_stop_option_code
Object Code	VAR
Data Type	INT16
Access	RW
PDO Mapping	NO
Units	--
Value Range	0,1,2,5,6
Default Value	0

value	Description
0	Power stage will be switched off. Motor is freely rotatable.。
1	Switch off the power stage after the motor stops deceleration.

2	Power stage will be shut down after the motor decelerates to still urgently.
5	QuickStop is alive after the motor decelerates to still.
6	QuickStop 状态 QuickStop is alive after the motor decelerates urgently to still.

6.2.6 halt_option_code

Halt_option_code determines how to stop when bit.8(halt) of controlword is set to 1.

Index	605D $_{h}$
Name	halt_option_code
Object Code	VAR
Data Type	INT16
Access	RW
PDO Mapping	NO
Units	--
Value Range	1,2
Default Value	0

Value	Description
1	The motor decelerates to still.
2	The motor decelerates urgently to still

6.2.7 fault_reaction_option_code

When an error is occurred, fault_reation_option_code determines how to stop.

Index	605D $_{h}$
Name	fault_reaction_option_code
Object Code	VAR
Data Type	INT16
Access	RW
PDO Mapping	NO
Units	--
Value Range	0
Default Value	0

Value	Description
0	Power stage will be switched off. Motor is freely rotatable.。。

7、Control mode

PRONET drive currently supports 3 control modes in CANopen DSP402：

HOMING MODE

PROFILE VELOCITY MODE

PROFILE POSITION MODE

This chapter mainly describes three control mode as above…

7.1 Relevant parameter of control mode

Index	Object	Name	Type	Attr.
6060 h	VAR	modes_of_operation	INT8	RW
6061 h	VAR	modes_of_operation_display	INT8	RO

7.1.1 modes_of_operation

Drive control mode will be determined by parameters in modes_of_operation.

Index	6060 h
Name	modes_of_operation
Object Code	VAR
Data Type	INT8
Access	RW
PDO Mapping	YES
Units	--
Value Range	1,3,6
Default Value	0

Value	Description
0	NOP MODE
1	PROFILE POSITION MODE
3	PROFILE VELOCITY MODE
6	HOMING MODE

7.1.2 modes_of_operation_display

Drive current control mode could be read from parameters in modes_of_operation_display …

Index	6061 **h**
Name	modes_of_operation_display
Object Code	VAR
Data Type	INT8
Access	RO
PDO Mapping	YES
Units	--
Value Range	1,3,6
Default Value	0

Notice： 1 、 The current control mode could be only known from parameters in modes_of_operation_display

2、Only when it is at the status of TargetReached, drive control mode could be switched form current mode to set control mode and then modes of operation_dsiplay could be identical to modes_of_operation.

7.2 HOMING MODE

PRONET servo drive currently supports multiple homing mode and users could choose the suitable homing mode. For example, if an incremental encoder is applied in servomotor, then homing mode of Zero impulse could be chosen and if serial encoder or resolver is applied in servomotor then Zero impulse homing mode couldn't be selected.

The user can determine the velocity, acceleration, and the kind of homing operation. After the
servo controller has found its reference the zero position can be moved to the desired point via the object home_offset （607C $_{h}$） .

7.2.1 Control word of homing mode.

15 ~ 9	8	7 ~ 5	4	3 ~ 0
*	Halt	*	home_start_operation	*

*: referred to previous chapters …

Name	Value	Description
Homing operation start	0	Homing mode inactive
	0 → 1	Start homing mode
	1	Homing mode active
	1 → 0	Interrupt homing mode
Halt	0	Execute the instruction of bit 4
	1	Stop axle with homing acceleration

7.2.2 Status word of homing mode

15 ~ 14	13	12	11	10	9 ~ 0
*	homing_error	homing_attained	*	target_reached	*

*: referred to previous chapters

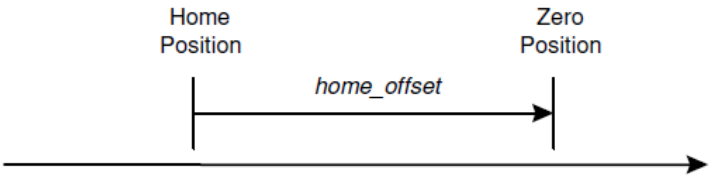
Name	Value	Description
Target reached	0	Halt = 0: Home position not reached Halt = 1: Axle decelerates
	1	Halt = 0: Home position reached Halt = 1: Axle has velocity 0
Homing attained	0	Homing mode not yet completed
	1	Homing mode carried out successfully
Homing error	0	No homing error
	1	Homing error occurred; Homing mode carried out not successfully; The error cause is found by reading the error code

7.2.3 Relevant parameter of homing mode

Index	Object	Name	Type	Attr.
607C h	VAR	home_offset	INT32	RW
6098 h	VAR	homing_method	INT8	RW
6099 h	ARRAY	homing_speeds	UINT32	RW
609A h	VAR	homing_acceleration	INT32	RW

home_offset

The object **home_offset** determines the displacement of the zero position to the limit resp. reference switch position.



Index	607C h
Name	home_offset
Object Code	VAR
Data Type	INT32
Access	RW
PDO Mapping	YES
Units	position units
Value Range	--
Default Value	0

homing_method

The negative and positive limit switch, the reference switch and the (periodic) zero impulse of the angle encoder.

Index	6098 h
Name	homing_method
Object Code	VAR
Data Type	INT8
Access	RW
PDO Mapping	YES
Units	--
Value Range	1,2,3,4,17,18,19,20
Default Value	1

Homing method value description

Value	Direction	Target	Reference point for Home position	DS402
1	Negative	NOT	Zero impulse	1

2	Positive	POT	Zero impulse	2
3	Negative	Reference switch	Zero impulse	3
4	Positive	Reference switch	Zero impulse	4
17	Negative	NOT	NOT	17
18	Positive	POT	POT	18
19	Negative	Reference switch	Reference switch	19
20	Positive	Reference switch	Reference switch	20

homing_speeds

There are two kinds of speeds required to find reference point, speed during search for switch and speed during search for zero.

Index	6099 **h**
Name	homing_speeds
Object Code	ARRAY
No. of Elements	2
Data Type	INT32

Sub-Index	01 **h**
Name	speed_during_search_for_switch
Object Code	VAR
Data Type	INT32
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	0

Sub-Index	02 **h**
Name	speed_during_search_for_zero
Object Code	VAR
Data Type	INT32
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	0

homing_acceleration

The objects **homing_acceleration** determine the acceleration which is used for all acceleration and deceleration operations during the search for reference.

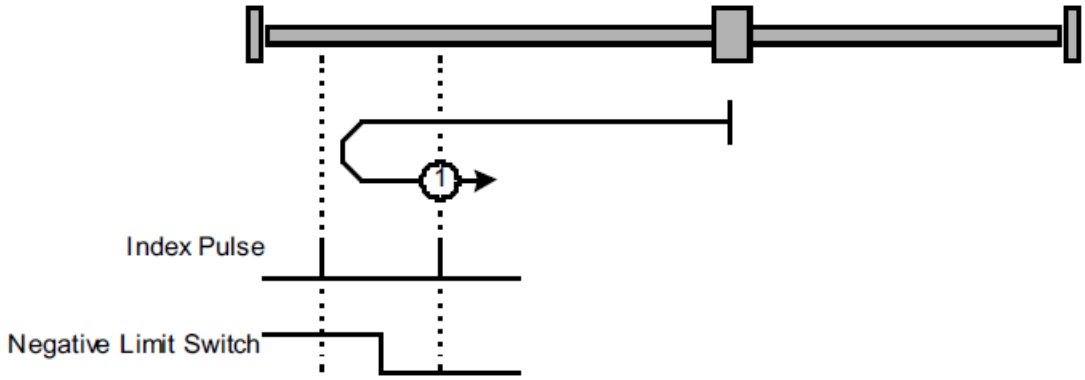
Index	609A **h**
Name	homing_acceleration
Object Code	VAR
Data Type	INT32
Access	RW

PDO Mapping	YES
Units	acceleration units
Value Range	--
Default Value	0

7.2.4 Homing sequences

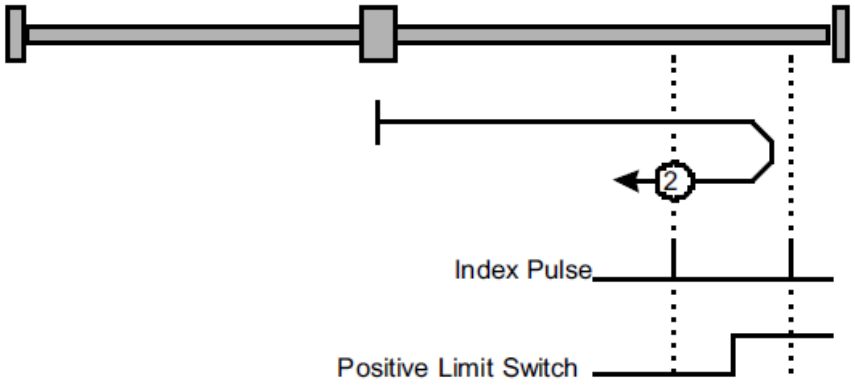
Method 1: Negative limit switch using zero impulse evaluation

If this method is used the drive first moves relatively quick into the negative direction until it reaches the negative limit switch. This is displayed in the diagram by the rising edge. Afterwards the drive slowly returns and searches for the exact position of the limit switch. The zero position refers to the first zero impulse of the angle encoder in positive direction from the limit switch.。



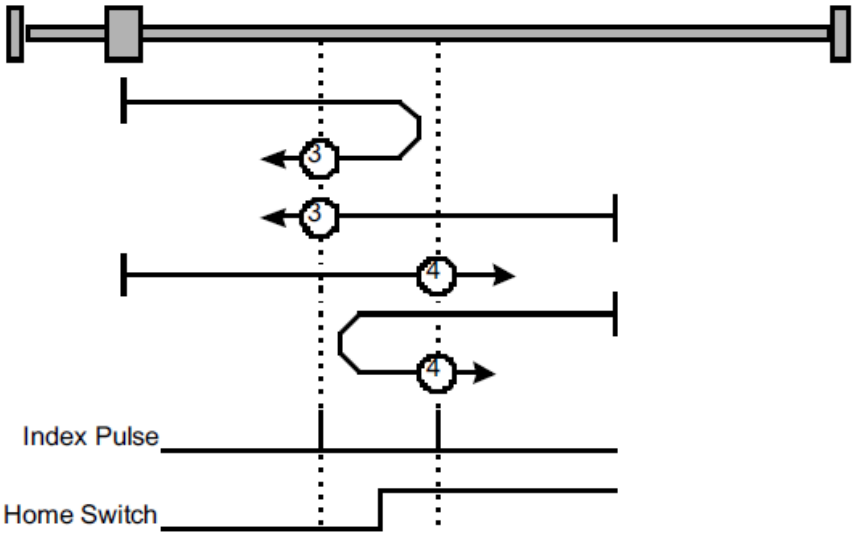
Method 2: Positive limit switch using zero impulse evaluation

If this method is used the drive first moves relatively quick into the positive direction until it reaches the positive limit switch. This is displayed in the diagram by the rising edge. Afterwards the drive slowly returns and searches for the exact position of the limit switch. The zero position refers to the first zero impulse of the angle encoder in negative direction from the limit switch.



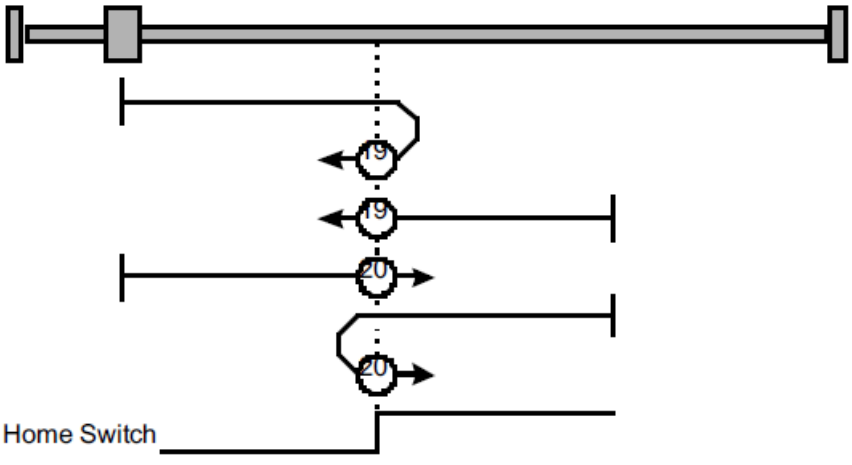
Methods 3 and 4: Reference switch and zero impulse evaluation

These two methods use the reference switch which is only active over parts of the distance. These reference methods are particularly useful for round-axis applications where the reference switch is activated once per revolution. In case of method 3 the drive first moves into positive and in case of method 4 into negative direction. Depending on the direction of the motion the zero position refers to the first zero impulse in negative or positive direction from the reference switch. This can be seen in the two following diagrams.



Method 17~20: Homing operation to the negative limit switch

If this method is used the drive first moves relatively quick into the negative direction until it reaches the negative limit switch. This is displayed in the diagram by the rising edge. Afterwards the drive slowly returns and searches for the exact position of the limit switch. The zero position refers to the descending edge from the negative limit switch.



7.3 PROFILE VELOCITY MODE

7.3.1 Control word of profile velocity mode

15 ~ 9	8	7 ~ 4	3 ~ 0
*	Halt	*	*

*: referred to previous chapters

Name	Value	Description
Halt	0	Execute the motion
	1	Stop axle

7.3.2 Status word of velocity mode

15 ~ 14	13	12	11	10	9 ~ 0
*	MaxSlippageError	Speed	*	Target reached	*

*: Referred to previous chapters

Name	Value	Description
Target reached	0	Halt = 0: *Target velocity* not (yet) reached Halt = 1: Axle decelerates
	1	Halt = 0: *Target velocity* reached Halt = 1: Axle has velocity 0
Speed	0	Speed is not equal 0
	1	Speed is equal 0
Max slippage error	0	Maximum slippage not reached
	1	Maximum slippage reached

7.3.3 Objects of profile velocity mode

Index	Object	Name	Type	Attr.
6069 $_h$	VAR	velocity_sensor_actual_value	INT32	RO
606B $_h$	VAR	velocity_demand_value	INT32	RO
606C $_h$	VAR	velocity_actual_value	INT32	RO
609D $_h$	VAR	velocity_window	UINT16	RW
606E $_h$	VAR	velocity_window_time	UINT16	RW
606F $_h$	VAR	velocity_threshold	UINT16	RW
6070 $_h$	VAR	velocity_threshold_time	UINT16	RW
60FF $_h$	VAR	target_velocity	INT32	RW

velocity_sensor_actual_value

The speed encoder is read via the object **velocity_sensor_actual_value**. The value is normalised in internal units. The velocity demand value can be read via this object.

Index	6069 **h**
Name	velocity_sensor_actual_value
Object Code	VAR
Data Type	INT32
Access	RW
PDO Mapping	YES
Units	0.1rmps (1R/10min)
Value Range	--
Default Value	--

velocity_demand_value

The velocity demand value can be read via this object. The unit of this object is the unit of user's speed unit. The velocity demand value can be read via this object.

Index	606B **h**
Name	velocity_demand_value
Object Code	VAR
Data Type	INT32
Access	RO
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	--

velocity_actual_value

The actual velocity value can be read via the object **velocity_actual_value**。The velocity demand value can be read via this object.

Index	606C **h**
Name	velocity_actual_value
Object Code	VAR
Data Type	INT32
Access	RO
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	--

velocity_window

With the object **velocity_window** a tolerance window for the velocity actual value will be defined for comparing the **velocity_actual_value**（606C $_h$）with the target velocity (**target_velocity** object **60FFh**). If the difference is smaller than the velocity window （606D $_h$） for a longer time than specified by the object **velocity_window_time** （606E $_h$） bit 10 (**target_reached)** will be set in the object **statusword**.

Index	606D $_h$
Name	velocity_window
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	20 R/10min

velocity_window_time

The object **velocity_window_time** serves besides the object **606Dh**: **velocity_window** to adjust the window comparator.

Index	606E $_h$
Name	velocity_window_time
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	ms
Value Range	--
Default Value	0

velocity_threshold

The object **velocity_threshold** determines the velocity underneath the axis is regarded as stationary. As soon as the **velocity_actual_value** exceeds the **velocity_threshold** longer than the **velocity_threshold_time** bit 12 is cleared in the **statusword**.

Index	606F $_h$
Name	velocity_threshold
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	10 R/10min

velocity_threshold_time

The object **velocity_threshold** determines the velocity below the axis is regarded as stationary. Its unit is ms。As soon as the **velocity_actual_value** exceeds the **velocity_threshold** longer than the **velocity_threshold_time** bit 12 is cleared in the **statusword**.

Index	6070 **h**
Name	velocity_threshold_time
Object Code	VAR
Data Type	UINT16
Access	RW
PDO Mapping	YES
Units	ms
Value Range	--
Default Value	0

target_velocity

The object **target_velocity** is the setpoint for the ramp generator.

Index	60FF **h**
Name	target_velocity
Object Code	VAR
Data Type	INT32
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	0

7.4 PROFILE POSITION MODE

7.4.1 Control word of profile position mode

15 ~ 9	8	7	6	5	4	3 ~ 0
*	Halt	*	abs / rel	Change set immediately	New set-point	*

*: referred to previous chapter

Name	Value	Description
New set-point	0	Does not assume *target position*
	1	Assume *target position*
Change set immediately	0	Finish the actual positioning and then start the next positioning
	1	Interrupt the actual positioning and start the next positioning
abs / rel	0	*Target position* is an absolute value
	1	*Target position* is a relative value
Halt	0	Execute positioning
	1	Stop axle with *profile deceleration* (if not supported with *profile acceleration*)

7.4.2 Status word of profile position mode

15 ~ 14	13	12	11	10	9 ~ 0
*	Following error	Set_point acknowledge	*	Target reached	*

*: referred to previous chapter

Name	Value	Description
Target reached	0	Halt = 0: *Target position* not reached Halt = 1: Axle decelerates
	1	Halt = 0: *Target position* reached Halt = 1: Velocity of axle is 0
Set-point acknowledge	0	Trajectory generator has not assumed the positioning values (yet)
	1	Trajectory generator has assumed the positioning values
Following error	0	No following error
	1	Following error

7.4.3 Objects of profile position mode

Index	Object	Name	Type	Attr.
607A $_{h}$	VAR	target_position	INT32	RW
6081 $_{h}$	VAR	profile_velocity	UINT32	RW
6082 $_{h}$	VAR	end_velocity	UINT32	RW
6083 $_{h}$	VAR	profile_acceleration	UINT32	RW
6084 $_{h}$	VAR	profile_deceleration	UINT32	RW
6085 $_{h}$	VAR	quick_stop_deceleration	UINT32	RW
6086 $_{h}$	VAR	motion_profile_type	INT16	RW

target_position

The object **target_position** determines the destination the servo controller moves to. The target position (**target_position**) is interpreted either as an absolute or relative position. This depends on bit 6 (**relative**) of the object **control word**.

Index	607A $_{h}$
Name	target_ position
Object Code	VAR
Data Type	INT32
Access	RW
PDO Mapping	YES
Units	position units
Value Range	--
Default Value	0

profile_velocity

The object **profile_velocity** specifies the speed that usually is reached during a positioning motion at the end of the acceleration ramp. The object **profile_velocity** is specified in **speed_units**.

Index	6081 $_{h}$
Name	profile_velocity
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	0

end_velocity

The object **end_velocity** defines the speed at the target position (**target_position**). Usually this object has to be set to zero so that the controller stops when it reaches the target position. For gapless sequences of positionings a value unequal zero can be set. The object **end_velocity** is specified in **speed_units** like the object **profile_velocity**.

Index	6082 **h**
Name	end_velocity
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	speed units
Value Range	--
Default Value	0

profile_acceleration

The object **profile_acceleration** determines the maximum acceleration used during a positioning motion. It is specified in user specific acceleration units (**acceleration_units)**.

Index	6083 **h**
Name	profile_acceleration
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	acceleration units
Value Range	--
Default Value	100000 R/10min/s

profile_deceleration

The object **profile_deceleration** specifies the maximum deceleration used during a positioning motion. This object is specified in the same units as the object **profile_acceleration**

Index	6084 **h**
Name	profile_deceleration
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	acceleration units
Value Range	--
Default Value	100000 R/10min/s

quick_stop_deceleration

The object **quick_stop_deceleration** determines the deceleration if a Quick Stop will be executed.

Index	6085 **h**
Name	quick_stop_deceleration
Object Code	VAR
Data Type	UINT32
Access	RW
PDO Mapping	YES
Units	acceleration units
Value Range	--
Default Value	200000 R/10min/s

motion_profile_type

The object **motion_profile_type** is used to select the kind of positioning profile. At present only a linear profile is available.

Index	6086 **h**
Name	motion_profile_type
Object Code	VAR
Data Type	INT16
Access	RW
PDO Mapping	YES
Units	--
Value Range	0
Default Value	0

7.4.4 Functional Description

Two different ways to apply target positions to the servo controller are supported.

Single setpoints

After reaching the **target_position** the servo controller signals this status to the host by the bit **target_reached** (Bit 10 of **controlword**) and then receives a new setpoint. The servo controller stops at the **target_position** before starting a move to the next setpoint.

Set of setpoints

After reaching the **target_position** the servo controller immediately processes the next **target_position** which results in a move where the velocity of the drive normally is not reduced to zero after reaching a setpoint.

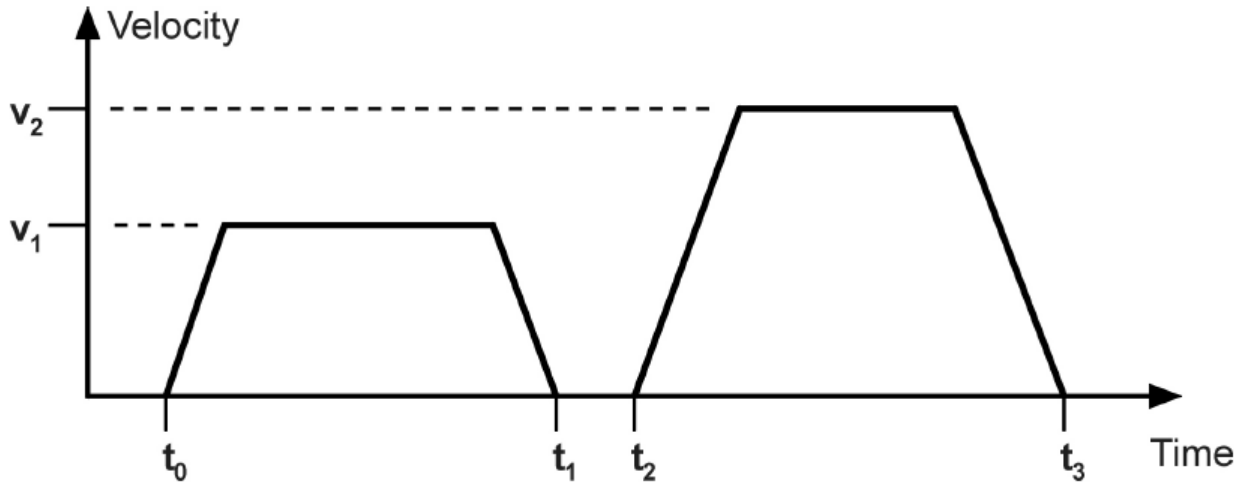
These Two methods are controlled by the bit4 and **bit5** in the object **controlword** and **set_point_acknowledge** in the object **statusword**. These bits are in a request-response relationship. So it is possible to prepare one positioning job while another job is still running.

Simple job positioning :

At first set NMT as Operational and control mode parameter (6061h) as 1.

1. At first the positioning data (**target_position:** 607A $_{h}$ **, profile_velocity, end_velocity** and **profile_acceleration**) are transferred to the servo controller.
2. The host can start the positioning motion by setting the bit4 (**new_set_point)** in the **controlword** as 1, bit5 (change_set_immediately) as 0 and bit6 as absolute or referential type according to target position type (absolute or referential).
3. This will be acknowledged by the servo controller by setting the bit **set_point_acknowledge** in the **statusword** when the positioning data has been copied into the internal buffer. Motion could be started now.

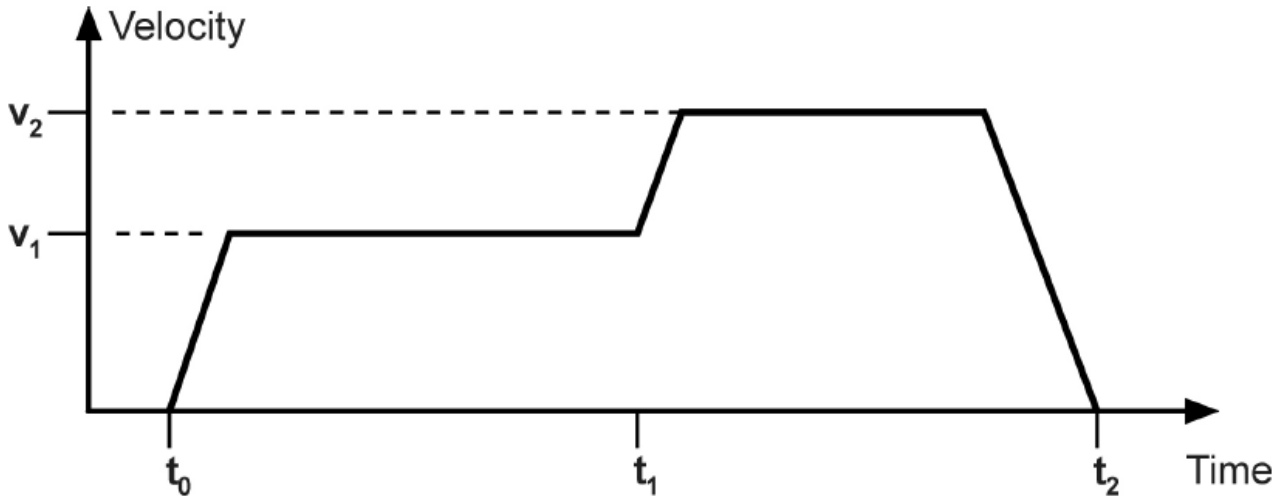
When the target is reached, drive will be acknowledged by bit 10 (target_reached) in status word. And then it will run gapless according to program or accept a new target position.



Gapless sequence of Positioning job :

At first set NMT as Operational and control mode parameter (6061h) as 1.

1. At first the positioning data (**target_position:** 607A**h, profile_velocity, end_velocity** and **profile_acceleration**) are transferred to the servo controller.
2. The host can start the positioning motion by setting the bit4 (**new_set_point)** in the **controlword** as 1, bit5 (change_set_immediately) as 0 and bit6 as absolute or referential type according to target position type (absolute or referential).
3. This will be acknowledged by the servo controller by setting the bit **set_point_acknowledge** in the **statusword** when the positioning data has been copied into the internal buffer. Motion could be started now.
4. Second positioning data (**target_position:** 607A**h, profile_velocity, end_velocity** and **profile_acceleration**) are transferred to the servo controller.
5. The host can start the positioning motion by setting the bit4 (**new_set_point)** in the **controlword** as 1, bit5 (change_set_immediately) as 0 and bit6 as absolute or referential type according to target position type (absolute or referential).
6. When the 1 target is reached driver will move forward to second target position. When the second target position is reached drive will be acknowledged by bit10 (target reached) in status word. And then it will be executed by program or accept another new target position.


Velocity
v2
v1
t0
t1
t2
Time

8、Parameters of the CAN interface

Parameter	Name and discription	Reboot required	Available for which control method	Functions and content
Pn006	**Hexadecimal**	required		
Pn703	**Hexadecimal**	required	ALL	**Pn703.0** CANopen **baud rate** [0] 50Kbps [1] 100Kbps [2] 125Kbps [3] 250Kbps [4] 500Kbps [5] 1Mbps **Pn703.1 Reserved for extension** **Pn703.2 Reserved for extension** **Pn703.3 Reserved for extension**
Pn704	**Axis address**	required	ALL	CANopen axis address

Appendix object dictionary

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
1000	--	VAR	device_type	UINT32	RO	NO	●				
1001	--	VAR	error_register	UINT8	RO	NO	●				
1003	--	VAR	pre_defined_error_field	UINT8	RW	NO	●				
1005	--	VAR	cob_id_sync	UINT32	RW	NO	●				
1006	--	VAR	communication_cycle_period	UINT32	RW	NO	●				
1007	--	VAR	synchronous_window_length	UINT32	RW	NO	●				
1008	--	VAR	manufacturer_device_name	STR	RO	NO	●				
1009	--	VAR	manufacturer_hardware_version	STR	RO	NO	●				
100A	--	VAR	manufacturer_software_version	STR	RO	NO	●				
1014	--	VAR	cob_id_emergency_message	UINT32	RW	NO	●				
1016	--	ARRAY	consumer_heartbeat_time	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		consumer_heartbeat_time1	UINT32	RW	NO	●				
1017		VAR	producer_heartbeat_time	UINT16	RW	NO	●				
1029	--	ARRAY	error_behaviour	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		communication_error	UINT8	RW	NO	●				
1200	--	RECORD	server_sdo_parameter	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		cob_id_client_server	UINT32	RO	NO	●				
	2		cob_id_server_client	UINT32	RO	NO	●				

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				unit
							All	PP	PV	HM	
1400	--	RECORD	receive_pdo_parameter_rpdo1	--	--	--	●				
	0		number_of_entries_rpdo1	UINT8	RO	NO	●				
	1		cob_id_used_by_pdo_rpdo1	UINT32	RO	NO	●				
	2		transmission_type_rpdo1	UINT8	RW	NO	●				
1401	--	RECORD	receive_pdo_parameter_rpdo2	--	--	--	●				
	0		number_of_entries_rpdo2	UINT8	RO	NO	●				
	1		cob_id_used_by_pdo_rpdo2	UINT32	RO	NO	●				
	2		transmission_type_rpdo2	UINT8	RW	NO	●				
1402	--	RECORD	receive_pdo_parameter_rpdo3	--	--	--	●				
	0		number_of_entries_rpdo3	UINT8	RO	NO	●				
	1		cob_id_used_by_pdo_rpdo3	UINT32	RO	NO	●				
	2		transmission_type_rpdo3	UINT8	RW	NO	●				
1403	--	RECORD	receive_pdo_parameter_rpdo4	--	--	--	●				
	0		number_of_entries_rpdo4	UINT8	RO	NO	●				
	1		cob_id_used_by_pdo_rpdo4	UINT32	RO	NO	●				
	2		transmission_type_rpdo4	UINT8	RW	NO	●				
1600	--	RECORD	receive_pdo_mapping_rpdo1	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_rpdo1	UINT32	RW	NO	●				
	2		second_mapped_object_rpdo1	UINT32	RW	NO	●				
	3		third_mapped_object_rpdo1	UINT32	RW	NO	●				
	4		fourth_mapped_object_rpdo1	UINT32	RW	NO	●				

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
1601	--	RECORD	receive_pdo_mapping_rpdo2	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_rpdo2	UINT32	RW	NO	●				
	2		second_mapped_object_rpdo2	UINT32	RW	NO	●				
	3		third_mapped_object_rpdo2	UINT32	RW	NO	●				
	4		fourth_mapped_object_rpdo2	UINT32	RW	NO	●				
1602	--	RECORD	receive_pdo_mapping_rpdo3	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_rpdo3	UINT32	RW	NO	●				
	2		second_mapped_object_rpdo3	UINT32	RW	NO	●				
	3		third_mapped_object_rpdo3	UINT32	RW	NO	●				
	4		fourth_mapped_object_rpdo3	UINT32	RW	NO	●				
1603	--	RECORD	receive_pdo_mapping_rpdo4	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_rpdo4	UINT32	RW	NO	●				
	2		second_mapped_object_rpdo4	UINT32	RW	NO	●				
	3		third_mapped_object_rpdo4	UINT32	RW	NO	●				
	4		fourth_mapped_object_rpdo4	UINT32	RW	NO	●				
1800	--	RECORD	transmit_pdo_parameter_tpdo1	--	--	--	●				
	0		number_of_entries_tpdo1	UINT32	RO	NO	●				
	1		cob_id_used_by_pdo_tpdo1	UINT32	RO	NO	●				
	2		transmission_type_tpdo1	UINT8	RW	NO	●				
	3		inhibit_time_tpdo1	UINT16	RW	NO	●				
	5		event_timer_tpdo1	UINT16	RW	NO	●				

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
1801	--	RECORD	transmit_pdo_parameter_tpdo2	--	--	--	●				
	0		number_of_entries_tpdo2	UINT32	RO	NO	●				
	1		cob_id_used_by_pdo_tpdo2	UINT32	RO	NO	●				
	2		transmission_type_tpdo2	UINT8	RW	NO	●				
	3		inhibit_time_tpdo2	UINT16	RW	NO	●				
	5		event_timer_tpdo2	UINT16	RW	NO	●				
1802	--	RECORD	transmit_pdo_parameter_tpdo3	--	--	--	●				
	0		number_of_entries_tpdo3	UINT32	RO	NO	●				
	1		cob_id_used_by_pdo_tpdo3	UINT32	RO	NO	●				
	2		transmission_type_tpdo3	UINT8	RW	NO	●				
	3		inhibit_time_tpdo3	UINT16	RW	NO	●				
	5		event_timer_tpdo3	UINT16	RW	NO	●				
1803	--	RECORD	transmit_pdo_parameter_tpdo4	--	--	--	●				
	0		number_of_entries_tpdo4	UINT32	RO	NO	●				
	1		cob_id_used_by_pdo_tpdo4	UINT32	RO	NO	●				
	2		transmission_type_tpdo4	UINT8	RW	NO	●				
	3		inhibit_time_tpdo4	UINT16	RW	NO	●				
	5		event_timer_tpdo4	UINT16	RW	NO	●				
1A00	--	RECORD	transmit_pdo_mapping_tpdo1	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_tpdo1	UINT32	RW	NO	●				
	2		second_mapped_object_tpdo1	UINT32	RW	NO	●				
	3		third_mapped_object_tpdo1	UINT32	RW	NO	●				
	4		fourth_mapped_object_tpdo1	UINT32	RW	NO	●				

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
1A01	--	RECORD	transmit_pdo_mapping_tpdo2	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_tpdo2	UINT32	RW	NO	●				
	2		second_mapped_object_tpdo2	UINT32	RW	NO	●				
	3		third_mapped_object_tpdo2	UINT32	RW	NO	●				
	4		fourth_mapped_object_tpdo2	UINT32	RW	NO	●				
1A02	--	RECORD	transmit_pdo_mapping_tpdo3	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_tpdo3	UINT32	RW	NO	●				
	2		second_mapped_object_tpdo3	UINT32	RW	NO	●				
	3		third_mapped_object_tpdo3	UINT32	RW	NO	●				
	4		fourth_mapped_object_tpdo3	UINT32	RW	NO	●				
1A03	--	RECORD	transmit_pdo_mapping_tpdo4	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		first_mapped_object_tpdo4	UINT32	RW	NO	●				
	2		second_mapped_object_tpdo4	UINT32	RW	NO	●				
	3		third_mapped_object_tpdo4	UINT32	RW	NO	●				
	4		fourth_mapped_object_tpdo4	UINT32	RW	NO	●				

Index	Subindex	Object	Name	Type	Attr.	PDO 映射	Support				Unit
							All	PP	PV	HM	
2000	--	RECORD	mask_tpdo1	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		mask1_tpdo1	UINT32	RW	NO	●				
	2		mask2_tpdo1	UINT32	RW	NO	●				
2001	--	RECORD	mask_tpdo2	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		mask1_tpdo2	UINT32	RW	NO	●				
	2		mask2_tpdo2	UINT32	RW	NO	●				
2002	--	RECORD	mask_tpdo3	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		mask1_tpdo3	UINT32	RW	NO	●				
	2		mask2_tpdo3	UINT32	RW	NO	●				
2003	--	RECORD	mask_tpdo4	--	--	--	●				
	0		number_of_entries	UINT8	RO	NO	●				
	1		mask1_tpdo4	UINT32	RW	NO	●				
	2		mask2_tpdo4	UINT32	RW	NO	●				

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
603F	--	VAR	error_code	UINT16	RW	YES	●				
6040	--	VAR	controlword	UINT16	RW	YES	●				
6041	--	VAR	statusword	UINT16	RO	YES	●				
605A	--	VAR	quick_stop_option_code	INT16	RW	NO	●				
605B	--	VAR	shutdown_option_code	INT16	RW	NO	●				
605C	--	VAR	disable_operation_option_code	INT16	RW	NO	●				
605D	--	VAR	stop_option_code	INT16	RW	NO	●				
605E	--	VAR	fault_reaction_option_code	UINT16	RW	NO	●				
6060	--	VAR	modes_of_operation	INT8	RW	YES	●				
6061	--	VAR	modes_of_operation_display	INT8	RO	YES	●				
6062	--	VAR	position_demand_value	INT32	RO	YES		●			position units
6063	--	VAR	position_actual_value*	INT32	RO	YES		●			inc
6064	--	VAR	position_actual_value	INT32	RO	YES		●			position units
6065	--	VAR	following_error_window	UINT32	RW	YES		●			position units
6066	--	VAR	following_error_time_out	UINT16	RW	YES		●			ms
6067	--	VAR	position_window	UINT32	RW	YES		●			position units
6068	--	VAR	position_window_time	UINT16	RW	YES		●			ms
6069	--	VAR	velocity_sensor_actual_value	UINT16	RW	YES			●		speed units
606B	--	VAR	velocity_demand_value	INT32	RO	YES			●		speed units
606C	--	VAR	velocity_actual_value	INT32	RO	YES			●		speed units
606D	--	VAR	velocity_window	UINT16	RW	YES			●		speed units
606E	--	VAR	velocity_window_time	UINT16	RW	YES			●		ms
606F	--	VAR	velocity_threshold	UINT16	RW	YES			●		speed units
6070	--	VAR	velocity_threshold_time	UINT16	RW	YES			●		ms
607A	--	VAR	target_position	INT32	RW	YES		●			position units

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
607B	--	ARRAY	position_range_limit	--	--	--		●			
	0		number_of_entries	UINT8	RW	NO		●			
	1		min_position_range_limit	INT32	RW	NO		●			position units
	2		max_position_range_limit	INT32	RW	NO		●			position units
607C	--	VAR	home_offset	INT32	RW	YES		●		●	position units
6081	--	VAR	profile_velocity	UINT32	RW	YES		●			speed units
6082	--	VAR	end_velocity	UINT32	RW	YES		●			speed units
6083	--	VAR	profile_acceleration	UINT32	RW	YES		●	●		acceleration units
6084	--	VAR	profile_deceleration	UINT32	RW	YES		●	●		acceleration units
6085	--	VAR	quick_stop_deceleration	UINT32	RW	YES		●	●		acceleration units
6086	--	VAR	motion_profile_type	INT16	RO	YES		●	●		
6093	--	ARRAY	position_factor	--	--	--		●		●	
	0		number_of_entries	UINT32	RW	NO		●		●	
	1		numerator	UINT32	RW	NO		●		●	
	2		divisor	UINT32	RW	NO		●		●	
6094	--	ARRAY	velocity_encoder_factor	--	--	--	●				
	0		number_of_entries	UINT32	RW	NO	●				
	1		numerator	UINT32	RW	NO	●				
	2		divisor	UINT32	RW	NO	●				
6097	--	ARRAY	acceleration_factor	--	--	--	●				
	0		number_of_entries	UINT32	RW	NO	●				
	1		numerator	UINT32	RW	NO	●				
	2		divisor	UINT32	RW	NO	●				
6098	--	VAR	homing_method	INT8	RW	YES				●	

Index	Subindex	Object	Name	Type	Attr.	PDO	Support				Unit
							All	PP	PV	HM	
6099	--	ARRAY	homing_speeds	--	--	--				●	speed units
	0		number_of_entries	UINT8	RW	YES				●	
	1		speed_during_search_for_switch	UINT32	RW	YES				●	speed units
	2		speed_during_search_for_zero	UINT32	RW	YES				●	speed units
609A	--	VAR	homing_acceleration	UINT32	RW	YES				●	acceleration units
60FA	--	VAR	control_effort	INT32	RO	YES		●			
60FC	--	VAR	position_demand_value*	INT32	RO	YES		●			inc
60FF	--	VAR	target_velocity	UINT32	RW	YES			●		speed units